Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Académico Profesional de Ingeniería de Sistemas

Razonamiento basado en casos (RBC) para toma de decisiones en proyectos de implementación de sistemas ERP utilizando jColibri

TESINA

Para optar el Título Profesional de Ingeniero de Sistemas

AUTORES

José Carlos COTRINA RAMOS

Daniel VERA QUINECHE

ASESOR

Virginia VERA POMALAZA

Lima, Perú

2009

Referencia bibliográfica

Cotrina, J. & Vera, D. (2009). *Razonamiento basado en casos (RBC) para toma de decisiones en proyectos de implementación de sistemas ERP utilizando jColibri*. Tesis para optar el título profesional de Ingeniero de Sistemas. Escuela Académico Profesional de Ingeniería de Sistemas, Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima, Perú.

Este trabajo está dedicado a Dios, quien siempre nos ilumina para seguir adelante.
A nuestras familias quienes siempre estuvieron con nosotros y nos dieron todo su amor y apoyo.

AGRADECIMIENTOS

Agradecemos de manera muy especial a nuestra Asesora: Virginia Vera Pomalaza, quien siempre nos ha motivado para poder culminar el presente trabajo y por compartir sus experiencias y conocimientos.

A los Profesores miembros del Jurado, por su orientación y dedicación para que este trabajo cumpla con los objetivos trazados.

Agradecemos a nuestros padres y hermanos quienes siempre han estado junto a nosotros brindándonos incondicionalmente su apoyo y comprensión a la largo del desarrollo de nuestra carrera.

Agradecemos a la empresa que nos ha permitido realizar el presente trabajo de investigación, por todo el apoyo que han podido brindarnos.

Agradecemos a todas aquellas personas quienes de una u otra manera siempre han estado apoyándonos moralmente para poder lograr la culminación de este trabajo.

A todas aquellas personas que indirectamente nos ayudaron a culminar este trabajo y que muchas veces constituyen un invalorable apoyo.

Y por encima de todo damos gracias a Dios.

Razonamiento Basado en Casos (RBC) para toma de decisiones en proyectos de implementación de Sistemas ERP utilizando jColibri

RESUMEN

Actualmente uno los principales problemas que enfrentan los proyectos de implementación de sistemas ERP es la correcta toma de decisiones, las cuales muchas veces no siguen una metodología y documentación adecuada, ya que este conocimiento persiste solo en la memoria de los expertos y no es reutilizada de forma eficiente a través de sistemas de información.

El presente trabajo propone un Modelo de Razonamiento Basado en Casos (RBC) que permita contar con información oportuna, ágil y confiable para la gestión del conocimiento y la toma de decisiones en proyectos de Implementación de Sistemas ERP.

Para realizar el presente trabajo se utilizará la herramienta jColibri la cual es un armazón o Framework orientado a objetos que facilita la construcción de sistemas de razonamiento basado en casos (RBC).

La implementación del modelo incrementa la calidad de la información y reduce los tiempos utilizados en solucionar los problemas de toma de decisiones de estos proyectos.

PALABRAS CLAVE: Gestión del Conocimiento

Razonamiento Basado en Casos (RBC)

Base de Casos

jColibri

Implementación de Sistemas ERP

CASE BASED REASONING (CBR) TO DECISION - MAKING IN ERP SYSTEM IMPLEMENTATION PROJECTS USING JCOLIBRI

ABSCTRACT

Currently one of the major problems that they confront the ERP systems implementation projects is the correct decisions take, which often do not follow an appropriate methodology and documentation, as this knowledge persists only the experts's memory and it is not reused of efficient form through information systems.

The present work proposes a Model of Case Based Reasoning (CBR) to allow for timely information, quick and reliable knowledge management and decision making in projects implemented ERP systems.

To make this work using the tool jColibri which is a frame or object-oriented framework that facilitates the construction of case-based reasoning (CBR).

The model's implementation increases information quality and reduce time spent on solving problems in decision-making of these projects.

PALABRAS CLAVE: Knowledge Management

Case Based Reasoning (CBR)

Base Case

jColibri

ERP System Implementation

INDICE

INDICE DE FIGURAS Y TABLAS

CAPITULO I

INTRODUCCION

1.1 Antecedentes

Las empresas actualmente están introduciendo una gran variedad de aplicaciones de TIC (Tecnología de información y comunicaciones), entre ellos los sistemas ERP (Entrerprise Resource Planing), muchas de ellas realizan grandes inversiones con valiosos datos incorporados, sin embargo a pesar de los beneficios que se puedan obtener, se sabe que hay un alto riesgo de fracaso en el proceso de Implementación. Los proyectos ERP de gran complejidad y riesgo necesitan de una buena combinación de conocimiento de negocio, tecnologías de información, servicios del fabricante del sistema y servicio de implantación de los consultores.

Mientras la industria ha experimentado significativos avances en la última década, hoy continúan produciéndose demasiados fracasos. Un informe publicado por el Standish Group en el año 2003 muestra que el 15% de los proyectos de tecnologías de la información fracasaron en el 2002, y el 51% adicional no consiguieron los resultados deseados en el tiempo o con el presupuesto previsto (Figura 1.1). Únicamente el 34% tuvieron éxito [72].

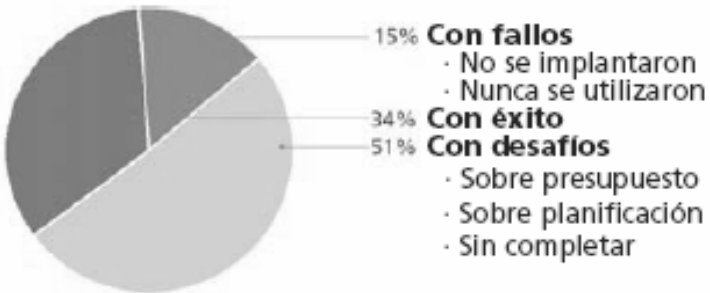


Fig. 1.1. Estadísticas de Implantaciones de Sistemas [72].

En el caso de las implantaciones más grandes y complejas de un sistema ERP, este porcentaje de éxito es aún menor.

Ana Juaristi Olalde menciona lo siguiente: “Como director de proyectos de ERP, puedo decir que en gran medida el éxito de la implantación no depende ni de la herramienta, ni del Cliente, ni de nuestra empresa. Depende directamente de nosotros los consultores” [71].

Por lo tanto puede considerarse como uno de los Principales motivos de fracaso en la implementación de un Sistema ERP que: Los consultores que realizan la implantación no están formados o no tienen la suficiente experiencia. Conocen lo básico de la herramienta, pero no son capaces de buscar soluciones alternativas. No tienen la suficiente experiencia para escuchar las necesidades del Cliente y buscar una posible configuración, una ruta alternativa en la operativa para solucionar aquel problema irresoluble que quizás con una puesta en común, una tormenta de ideas o la aportación de un consultor más experimentado se solucionaría [71].

En ocasiones, cuando el mercado es favorable y se producen varias ventas del ERP en el mismo periodo de tiempo, la empresa implantadora se encuentra con varios Clientes que solicitan comenzar la implantación lo antes posible. Y se presentan algunos inconvenientes porque los consultores experimentados son habitualmente muy solicitados y su agenda suele estar bastante apretada. La solución suele ser, contratar consultores sin experiencia en la herramienta, que difícilmente van a buscar soluciones al Cliente. Si además, por ahorrar costes, estos consultores son recién licenciados o tienen mínima experiencia previa con otras herramientas, es imposible que funcione. El cliente se sentirá decepcionado. Pasarán las jornadas de formación totalmente insatisfactorias. Al final un consultor experimentado tendrá que recuperar la cuenta, si es que aún es posible recuperarla por lo cual se generaran problemas para todos [71].

Por lo tanto, el éxito o fracaso de la implementación de un ERP depende de contar con los recursos técnicos y humanos adecuado además de considerar una administración efectiva del proyecto. Es imprescindible contar con un Consultor con experiencia en este tipo de proyectos que sepa manejar las situaciones políticas, de riesgo, tecnológicas, de conflicto y de comunicación inherentes y que sepa comunicar y capacitar adecuadamente sus conocimientos al personal de la corporación.

Para garantizar el éxito de la implementación se necesita contar con asesoría de expertos para la resolución de cualquier problema y dar seguimiento a todos los procesos. El conocimiento o la experiencia previa son fundamentales para la toma de decisiones ante cualquier problema o situación durante el proyecto. La falta de experiencia o desconocimiento puede llevar a tomar decisiones equivocadas o erróneas, ya que no se tuvieron las suficientes herramientas para realizar una adecuada toma de decisión.

Debido a la naturaleza del proyecto, es importante contar con el personal técnico debidamente capacitado. Un buen equipo técnico, con personal experimentado en ERP, es definitivamente difícil de conseguir. El personal valioso de una empresa es requerido en varios proyectos, lo que hace necesario identificar si se cuenta o no con las personas críticas desde unos principios mismos que deben asegurar su permanencia a través de toda la implementación. Si se tiene la experiencia de haber estado en implantaciones similares exitosas, el consultor podrá visualizar estos casos, casi siempre sin ningún problema.

Debido a la importancia de las tomas de decisiones en los proyectos de implementación de sistemas ERP, estas son tomadas por los encargados de los proyectos, consultando su experiencia anterior ante situaciones similares o consultando a personal experto en el tema, por lo cual esta tarea se realiza de forma manual sin apoyo en ninguna herramienta que pueda facilitar el análisis de la situación.

Sin embargo, no se ha llegado a implementar un sistema que permita dar una solución de mayor eficiencia, que garantice el conocimiento y experiencia en los temas consultados.

Esto se origina por el gran volumen de información que no está organizada en bases de datos automatizadas y accesibles a las personas interesadas.

Por ello, consideramos que las empresas encargadas de implementar sistemas ERP hoy en día deben ser capaces de manejar información valiosa, detallada de los datos para a través de esos datos obtener un conocimiento más valioso y por ende poder tomar las mejores decisiones de acuerdo a la situación o caso presentado.

1.2 Formulación del Problema

La Toma de Decisiones dentro de los Proyectos de Implementación de Sistemas ERP es uno de los procesos de mayor importancia dentro del ciclo de implementación, en ella se determina cual es la decisión que se tomará para una determinada situación considerando las características y criterios que se deban tomar para un determinado caso. Este proceso esta a cargo de las personas que Gestionan el Proyecto, así como también por los consultores encargados del Proyecto, y cuya función es determinar las mejores alternativas de solución para un determinado caso, para ello es necesario realizar un análisis cuidadoso de los diversos puntos tanto cualitativos como cuantitativos, que en conjunto permitirá tener una mejor visión sobre el problema y la capacidad para determinar la mejor solución.

Actualmente la Mayoría de Empresas que se dedican a implementación de Sistemas ERP no cuentan con una herramienta de apoyo para la toma de decisiones que les permita gestionar el conocimiento de forma rápida y eficaz.

Es claro que el conocimiento de la aplicación debe ser compartido entre los miembros que participan en el desarrollo del proyecto, sin importar si hacen parte o no del grupo de implementación. Desde el cliente hasta el personal que realiza las pruebas deben tener en común el conocimiento que hace parte de la solución que están desarrollando. Su interpretación debe ser la misma, si no se corre el riesgo de empezar a hacer supuestos erróneos. Por esta razón se debe integrar diferentes fuentes de conocimiento para lograr desarrollar el proyecto correctamente [73].

Se puede ver al conocimiento como un concepto fundamental dentro de las organizaciones, ya que puede potencializar los servicios de tecnología de información y los sistemas de soporte dentro de un proyecto. La idea a desarrollar es simple: dar a los desarrolladores el conocimiento preciso y actualizado que ellos necesitan, para solucionar problemas encontrados en un sistema, y mantener el entorno de las tecnologías de información de este campo ejecutándose eficientemente. Para lograr esta meta, se debe coleccionar los problemas

y soluciones obtenidas a lo largo de los proyectos pasados para construir una base de conocimiento [73].

La falta de administración del conocimiento ha limitado a la empresa poder dar solución inmediata a problemas frecuentes en la operación, ocasionando demoras o defectos al momento de desarrollar el producto o servicio. Asimismo la carencia de manuales que faciliten el entendimiento del desarrollo de las actividades de la organización, ha generado dificultades en el rendimiento y pérdidas aun no cuantificadas, como lo son: costo de perder un cliente al no cubrir sus necesidades, al realizar correcciones en los servicios entregados por no cumplir con las especificaciones, entre otras [73].

Del mismo modo, la deficiencia en la disponibilidad del conocimiento limita a la organización el poder sustentar las decisiones al momento de realizar ofertas en productos y servicios; además de generar esfuerzos innecesarios por obtener información cada vez que se necesita y que este flujo sea lento, orillando a dejar de aprovechar las oportunidades que se presentan, reflejando una falta de profesionalismo [73].

Para el Método Case Based Reasoning (CBR), se recomienda su implementación en el área operativa, este método parte de la relación causa-efecto. Surge de hechos pasados, y que al presentarse nuevamente el problema en la producción se aplique la solución predeterminada en base a experiencias pasadas. Por lo tanto puede ser una excelente herramienta para los trabajadores en la solución de problemas frecuentes y no tener que esperar al experto.

1.3 Objetivos

1.3.1 Objetivos Generales

Diseñar una Herramienta de Razonamiento Basado en Casos (RBC) para toma de decisiones que permita alcanzar los siguientes Objetivos Generales:

Contar con información oportuna, ágil y confiable para la gestión del conocimiento y la toma de decisiones en proyectos de Implementación de Sistemas ERP; de manera que permita incrementar la calidad de la información y reducir los tiempos utilizados en solucionar los problemas de toma de decisiones en proyectos de Implementación de Sistemas ERP.

1.3.2 Objetivos Específicos

Asimismo, el diseño de este sistema plantea los siguientes objetivos específicos:

- Modelar el conocimiento de la Organización planteando una base de casos de opiniones validas que permita generar precedentes en futuros casos similares al analizado.
- Lograr resaltar la importancia de la Gestión del conocimiento en las Organizaciones.
- Lograr la adquisición de conocimiento para la Organización.

- Extraer conocimiento de manera más fácil mediante casos ya ocurridos en experiencias anteriores.

- Desarrollar un prototipo que implemente este modelo y valide las teorías y objetivos propuestos.

1.4 Justificación

El tema de investigación abarca un problema relevante en las empresas que realizan trabajos de implementación de sistemas ERP, debido a que no se gestiona el conocimiento eficientemente ya que se presentan dificultades de toma de decisiones evaluando experiencias pasadas.

La Importancia del proceso de toma de decisiones dentro de los proyectos de implementación radica en que está puede determinar el éxito o fracaso dentro de un proyecto y para esto se necesita un modelo que permita construir una herramienta de apoyo para la toma de decisiones. Y además, toda empresa que desea adquirir ventajas competitivas sobre sus competidores y el conocimiento que posee es un recurso de mucha importancia en la sociedad en que vivimos pues es una sociedad del conocimiento y quien mejor lo gestione logrará mantenerse en el mercado actual.

Se hace necesario, entonces, combinar herramientas con la experiencia de los especialistas para afrontar los nuevos retos que impone diseñar sistemas eficientes y novedosos en las condiciones actuales, sumamente cambiantes. Esta necesidad es aún más imperiosa cuando cada individuo en una organización tiene su propia forma de abordar los problemas y no siempre tiene todo el conocimiento y experiencia necesarios sobre el negocio.

Contar con una base de ejemplos que ayude en la toma de decisiones en proyectos de implementación de Sistemas ERP basada en la experiencia de otros especialistas que han abordado proyectos con características similares, mejoraría considerablemente la calidad del producto final y permitiría a los consultores y jefes de proyecto hacer sugerencias que contribuyan a mejorar los procesos productivos y de negocios para los usuarios del sistema.

Por lo tanto el presente trabajo tiene su justificación en la complejidad que presenta determinar la mejor decisión para resolver el problema debido a que, para este tipo de evaluación se toman en cuenta muchos factores y variantes del problema, para lo cual se necesita evaluar situaciones similares ocurridas en el pasado (casos), para encontrar soluciones a los mismos, modificar soluciones existentes y explicar situaciones anómalas. Los sistemas de Razonamiento Basado en Casos (CBR), son capaces de utilizar conocimiento específico de experiencias previas para resolver un problema, capturan las características de dicho problema, buscan casos históricos con valores similares para dichas características de dicho problema, analizan las soluciones de estos casos y proponen una solución al problema, y finalmente, aprenden del problema actual para problemas futuros.

1.5 Alcances y Limitaciones

Ante la necesidad de las empresas dedicadas a implementar sistemas ERP de tomar decisiones eficientes ante los problemas o situaciones que se presenten en los proyectos, se hace pertinente que estas se encuentren apoyada por herramientas que le permitan dar solidez a la

decisión tomada por parte del personal encargado de dirigir el proyecto, tanto de los encargados de la gestión del proyecto así como los encargados de implementar el sistema ERP en las empresas, adaptándose a las características particulares de cada caso y muchas veces partiendo de información imprecisa e insuficiente, por lo que usar procedimientos manuales no es suficiente para obtener resultados significativos.

En este trabajo se presenta el modelo de un Sistema de Razonamiento Basado en Casos para la toma de decisiones en los proyectos de implementación de herramientas ERP. Específicamente se trabajará con la solución Oracle E-Business Suite, y los módulos que serán abordados serán los financieros.

Con esto lo que se busca es minimizar el riesgo de las decisiones tomadas y agilizar dicho proceso basándolo en experiencias anteriores (casos). Cabe resaltar que el modelo implementado sólo será aplicable a la toma de decisiones en los proyectos de implementación de sistemas ERP. Para la aplicación práctica, se diseñará un prototipo que simule dicho modelo en una empresa dedicada a la implementación de la solución ERP especificada.

1.6 Propuesta

El proceso propuesto sigue una estrategia de administración de conocimiento, el cual utiliza los conceptos siguientes: la obtención, el almacenamiento, y el uso del conocimiento. Específicamente la estrategia se basa en el razonamiento por analogías, también llamado razonamiento basado en casos (CBR).

En el razonamiento basado en casos, cada caso es la descripción, por medio de características, de un objeto de conocimiento que es de interés mantener y administrar. En nuestro modelo, cada caso corresponde a un caso o problema presentado en el proceso de toma de decisiones en proyectos de implementación de Sistemas ERP. En este contexto, por medio del CBR se puede intentar predecir las soluciones para nuevos casos o problemas, basados en un conjunto de datos históricos que se describe por medio de un conjunto de atributos. La solución del caso o problema a ser encontrada es prevista utilizando la información de los casos más similares o analogías más cercanas.

En el contexto de toma de decisiones en proyectos de implementación de Sistemas ERP, las ontologías sirven como fuentes de conocimiento para alcanzar métodos de razonamiento semántico, ayudando a valorar la similitud y la adaptación de las soluciones que pueden ser re-usables a través de diferentes dominios. Además, la ontología ayuda a identificar el contexto semántico del mensaje de reporte de un caso o problema, y permite la asociación de los casos por medio del contenido semántico. Esto trae como ventaja que se puede enriquecer la descripción de un caso o problema de toma de decisiones en proyectos de implementación de Sistemas ERP para ampliar el conocimiento de este dominio.

Ante la necesidad de las instituciones que se dedican a implantaciones de sistemas ERP, de mejorar sus procesos de toma de decisiones para garantizar el éxito de dichos proyectos, se hace pertinente el uso de una herramienta de apoyo para estas acciones críticas.

En el presente trabajo se presenta un prototipo de sistema utilizando la técnica de Razonamiento Basado en Casos (RBC), el cual permitirá realizar un proceso de toma de

decisiones más eficientes, con lo cual se reduce los riesgos que pongan en peligro el éxito de la implantación. Para la elaboración del sistema se utilizará el framework jCOLIBRI, realizado en java, el cual permite crear sistemas de RBC de forma sencilla, rápida y eficiente.

El caso de aplicación será realizado en la empresa HSP S.A.C., la cual se dedica a la implantación del ERP ORACLE e-bussines en empresas del sector peruano. Cabe resaltar que el prototipo estará orientado a la solución de este tipo de proyectos de implantación.

1.7 Organización de la Tesina

La organización de la tesina consta de 6 capítulos.

El primer capítulo está dedicado a la presentación del problema, los objetivos de la investigación, así como la justificación, propuesta y alcances.

En el segundo capítulo, el conocimiento necesario para llevar a cabo el estudio.

En el tercer capítulo, se presenta las soluciones alternativas que presentan diferentes organizaciones para resolver problemas similares al de estudio de este trabajo.

En el cuarto capítulo, se presenta la empresa donde será aplicado el trabajo de investigación, una evaluación comparativa entre Frameworks CBR utilizados para este tipo de problemas, la solución del problema explicando el método y herramientas seleccionadas para la implementación de la solución.

En el quinto capítulo, se presenta la herramienta propuesta para la resolución del problema según la metodología presentada en el capítulo cuarto.

En el sexto capítulo, se presenta las conclusiones a las que se llegaron, así como también el listado de los posibles trabajos futuros a realizar.

CAPITULO II

MARCO TEORICO

2.1 Introducción

El propósito de este capítulo es entregar una revisión de los conceptos básicos en relación al tema de investigación de la presente tesina. Básicamente, los conceptos fundamentales que serán abordados en este capítulo son, Gestión de conocimiento, Sistemas Expertos, Razonamiento basado en casos, jCOLIBRI y se dará una descripción de la Organización donde se aplicará el caso práctico. Se consideran estas definiciones relevantes y necesarias para el entendimiento tanto del problema como de la solución propuesta.

2.2 Gestión del Conocimiento

Fernando Sáez Vacas [18], define la gestión del conocimiento como el proceso de identificar, agrupar, ordenar y compartir continuamente conocimiento de todo tipo para satisfacer necesidades presentes y futuras, para identificar y explotar recursos de conocimiento tanto existentes como adquiridos y para desarrollar nuevas oportunidades.

La práctica habitual es crear un foro virtual donde las experiencias individuales y los conocimientos se suman en un espacio que pueda ser accesible a todos sus miembros [18].

Algunas de las razones que justifican la aparición de estos procesos de gestión del conocimiento [18].

- El mercado es cada vez más competitivo, lo que demanda mayor innovación en los productos. Debido a esto, el conocimiento debe desarrollarse y ser asimilado cada vez con mayor rapidez.

- Las empresas están organizando sus negocios enfocando sus esfuerzos en crear mayor valor para sus clientes. Los niveles administrativos se han ido reduciendo. Existe la necesidad de reemplazar la manera informal en la que se gerencia el conocimiento en las funciones administrativas por métodos formales dentro de procesos de negocios orientados al cliente.

- La presión de la competencia está reduciendo el tamaño de los grupos de empleados que poseen el conocimiento de la empresa.
- Se requiere tiempo para adquirir conocimiento y lograr experiencia a partir de él. Los empleados cada vez tienen menos tiempo para hacer esto.
- Está creciendo la tendencia dentro de los empleados de retirarse cada vez más temprano en su vida laboral o de aumentar su movilidad entre empresas, lo cual ocasiona que el conocimiento se pierda.
- Existe la necesidad de manejar cada vez mayor complejidad en empresas pequeñas y con operaciones transnacionales.

Algunos objetivos de la Gestión del conocimiento [19]:

- Formular una estrategia de alcance organizacional para el desarrollo, adquisición y aplicación del conocimiento.
- Implantar estrategias orientadas al conocimiento.
- Promover la mejora continua de los procesos de negocio, enfatizando la generación y utilización del conocimiento.
- Monitorear y evaluar los logros obtenidos mediante la aplicación del conocimiento.
- Reducir los tiempos de ciclos en el desarrollo de nuevos productos, mejoras de los ya existentes y la reducción del desarrollo de soluciones a los problemas.
- Reducir los costos asociados a la repetición de errores.

La Gestión del Conocimiento está asociada al proceso sistémico de administración de la información [19]. Este proceso se puede apreciar en la Figura 2.1.

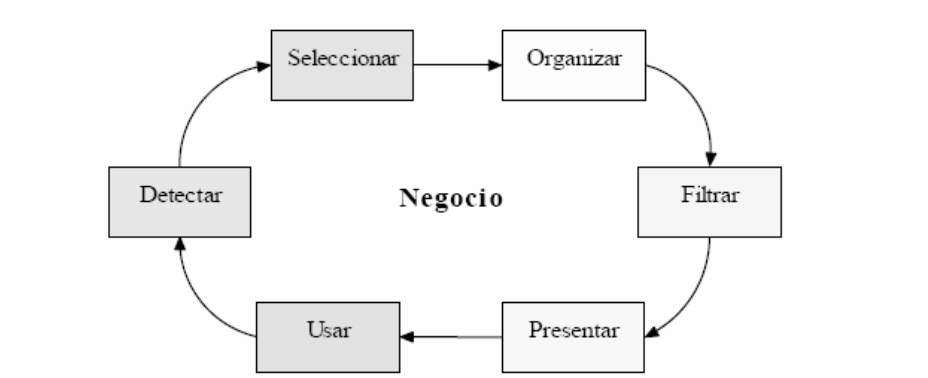


Fig. 2.1. Proceso de Gestión del Conocimiento [19].

Donde: [19]

- **Detectar:** Es el proceso de localizar modelos cognitivos y activos (pensamiento y acción) de valor para la organización, el cual radica en las personas. Son ellas, de acuerdo a sus capacidades cognitivas (modelos mentales, visión sistémica, etc.), quienes determinan las nuevas fuentes de conocimiento de acción. La fuentes de

conocimiento pueden ser generadas tanto de forma interna (I&D, proyectos, descubrimientos, etc.) como externa (fuentes de información periódica, Internet, cursos de capacitación, libros, etc.).

- **Seleccionar:** Es el proceso de evaluación y elección del modelo en torno a un criterio de interés. Los criterios pueden estar basados en criterios organizacionales, comunales o individuales, los cuales estarán divididos en tres grandes grupos: Interés, Práctica y Acción. Sería ideal que la o las personas que detectaron el modelo estuvieran capacitada y autorizadas para evaluarla, ya que esto permite distribuir y escalar la tarea de seleccionar nuevos modelos. En todo caso deberán existir instancias de apoyo a la valoración de una nueva fuente potencial.

- **Organizar:** Es el proceso de almacenar de forma estructurada la representación explícita del modelo. Este proceso se divide en las siguientes etapas:

    - **Generación:** Es la creación de nuevas ideas, el reconocimiento de nuevos patrones, la síntesis de disciplinas separadas, y el desarrollo de nuevos procesos.

    - **Codificación:** Es la representación del conocimiento para que pueda ser accedido y transferido por cualquier miembro de la organización a través de algún lenguaje de representación (palabras, diagramas, estructuras, etc.). Cabe destacar que la representación de codificación puede diferir de la representación de almacenamiento, dado que enfrentan objetivos diferentes: personas y máquinas.

    - **Trasferencia:** Es establecer el almacenamiento y la apertura que tendrá el conocimiento, ayudado por interfaces de acceso masivo (por ejemplo, la Internet o una Intranet), junto de establecer los criterios de seguridad y acceso. Además debe considerar aspectos tales como las barreras de tipo Temporales (Vencimiento), de Distancias y Sociales.

- **Filtrar:** Una vez organizada la fuente, puede ser accedida a través de consultas automatizadas en torno a motores de búsquedas. Las búsquedas se basarán en estructuras de acceso simples y complejas, tales como mapas de conocimientos, portales de conocimiento o agentes inteligentes.

- **Presentar:** Los resultados obtenidos del proceso de filtrado deben ser presentados a personas o máquinas. En caso que sean personas, las interfaces deben estar diseñadas para abarcar el amplio rango de comprensión humana. En el caso que la comunicación se desarrolle entre máquinas, las interfaces deben cumplir todas las condiciones propias de un protocolo o interfaz de comunicación.

- **Usar:** El uso del conocimiento reside en el acto de aplicarlo al problema objeto de resolver. De acuerdo con esta acción es que es posible evaluar la utilidad de la fuente de conocimiento a través de una actividad de retroalimentación.

Entre los modelos de gestión del conocimiento, el más conocido y aceptado es el de Nonaka y Takeuchi [18] que diferencian dos tipos de conocimiento:

- **Conocimiento Explícito:** se trata del conocimiento basado en datos concretos que pueden ser expresados en lenguaje formal y por lo tanto es empaquetable. Puede utilizarse y compartirse empleando algún medio conveniente. Es transferible, siempre que el receptor posea las claves de conocimiento adecuadas para aprovecharlo. Por Ejemplo: formulas, ecuaciones, software, tecnología en general.

- **Conocimiento Tácito:** parte del conocimiento que es específico del contexto, es personal y difícil de formalizar, comunicar y transferir. Se compone de ideas, habilidades y valores del individuo. Está íntimamente ligado a las personas determinando sus conductas. No está registrado por ningún medio, por ello es más difícil de compartir. Por Ejemplo: el know-how, los modelos mentales y las experiencias.

En la Figura 2.2., se puede observar las principales formas de transformación de los tipos de conocimiento. Los cuatro procesos de conversión del conocimiento son los siguientes: [18]

1. **Tácito a Tácito (socialización).** Los individuos adquieren nuevos conocimientos directamente de otros.

2. **Tácito a Explícito (externalización).** El conocimiento se articula de una manera tangible, a través del dialogo, plasmándose en esquemas, formulas y métodos.

3. **Explícito a Explícito (combinación).** Se combinan diferentes formas de conocimiento explícito mediante documentos o bases de datos.

4. **Explícito a Tácito (internalización).** Los individuos internalizan el conocimiento de los documentos en su propia experiencia.

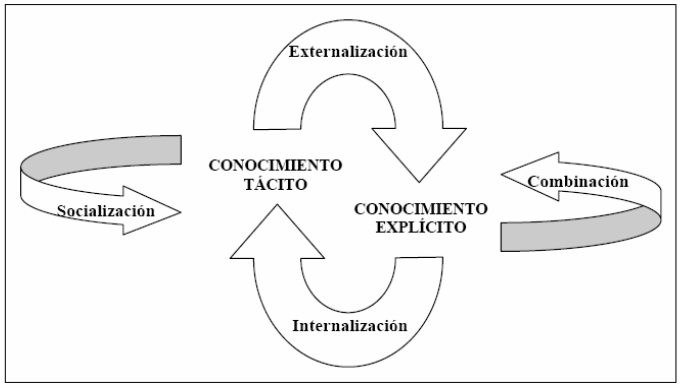


Fig. 2.2. Los Cuatro Procesos de Conversión del Conocimiento [18].

Según Sáez Vacas [18], "Las Nuevas tecnologías de la información y las comunicaciones han permitido que la transmisión y gestión de conocimiento pueda ser una realidad. Así, si consideramos una cadena formada por: datos-información-conocimiento, la última dimensión surge de la gestión eficaz de las dos anteriores; aunque, tampoco hay información sin conocimiento y sin esfuerzo, lo que significa que una información que llega a un receptor

humano desprovisto del conocimiento y del lenguaje pertinentes –las claves- es clasificada como ruido".

Datos	Información	Conocimiento
Observaciones sencillas de distintos sucesos. • Se capturan con facilidad en las maquinas. • Se estructuran fácilmente. • A menudo se cuantifican. • Se transfieren con facilidad.	Datos dotados de pertinencia y propósito. • Requiere una unidad de análisis. • Necesita consenso sobre el significado. • La intermediación humana es indispensable.	Información valiosa de la mente humana. • Difícil de estructurar. • Difícil de capturar en las máquinas. • A menudo es tácito. • La transferencia es complicada.

Tabla 2.1. Características Dato, Información, Conocimiento.

Según Sáez Vacas [18], "Lo esencial no es la información, sino la cantidad y clase de conocimiento que ésta contiene. Siempre que nos refiramos a procesos cognitivos, no a mera comunicación social, es preciso admitir la supremacía del conocimiento sobre la información. No obstante, debemos hacer algunas matizaciones".

- No hay conocimiento sin información y sin trabajo para procesarla.
- No hay información sin conocimiento y sin trabajo.
- El aumento de información incrementa el conocimiento, proceso sometido en uno u otro momento a una ley de rendimientos decrecientes.
- El aumento de conocimiento incrementa la eficacia del procesamiento de información.
- Cualquier acción meritoria se construye a través del conocimiento y más trabajo.
- Un exceso de información tiende a anular la creación o regeneración del conocimiento.

Según Sáez Vacas [18], "En el plano cuantitativo, a mayor cantidad y densidad de información, mayor necesidad de esfuerzo para construir conocimiento, mientras que, en sentido contrario, cuanto mayor es el conocimiento del receptor menor es el esfuerzo para procesar una determinada cantidad de información".

Según Sáez Vacas [18], "El grado de participación humana aumenta a medida que se avanza por esta cadena (datos-información-conocimiento). La Infotecnología incide especialmente sobre las dos primeras dimensiones y contribuye a facilitar la generación de la tercera, pero no son suficientes, pues los ordenadores son idóneos para manejar datos, menos aptos para la información y mucho menos para el conocimiento. Es por ello que, en una trilogía personas-

procesos-tecnología, la gestión del conocimiento pone énfasis en las personas como procesadoras de símbolos para generar nuevos significados y procesos, dejando a las nuevas tecnologías como una herramienta necesaria pero relegada al último lugar".

En palabras de Sáez Vacas [18], "Cometeríamos pecado de ingenuidad suponiendo que la tecnología, con su sola presencia a nuestro lado, nos hace más inteligentes. El juego no funciona así, sino de esta otra forma: para ser útil, eficaz o liberadora, según los casos, la tecnología de la información, especialmente cuanto más avanzada sea, como la informática y técnicas afines, exige más inteligencia, prudencia, y hasta sabiduría. Asimilar, comprender y organizar adecuadamente estos paquetes de conocimiento requiere esfuerzos notables de formación, reflexión y experiencia".

2.3 Sistemas Expertos

Castillo, Gutiérrez y Hadi [6], definen un sistema experto como "Un sistema informático (hardware y software) que simula a los expertos humanos en un área de especialización dada".

Para que un sistema experto sea una herramienta útil y efectiva tanto en la propia interacción con el usuario como la calidad de la solución dada, el sistema experto tendría que reunir dos características claramente diferenciadas y fundamentales, de un lado debería de ser capaz de explicar sus propios razonamiento, es decir, aquel conjunto de reglas o pasos que utiliza el experto e ir deduciendo diferentes hechos hasta llegar a una conclusión final. Por otro lado, un sistema experto, al igual que el ser humano, debe ser capaz de adquirir nuevos conocimientos, ya que sin esto puede que consiguiésemos un sistema de una gran calidad en la actualidad, pero la gran velocidad a la que avanza el mundo actualmente haría que rápidamente este se quedase obsoleto y por lo tanto resultará inútil. Para solucionar estos problemas los expertos recurren a mecanismos de razonamiento que sirven para modificar los conocimientos anteriores. Viendo esto podemos llegar a la conclusión de que los sistemas expertos, en muchos campos no intentan substituir a los expertos humanos, sino que lo que se persigue es realizar con más rapidez y eficacia todas las tareas que realiza. Debido a esto, en la actualidad los sistemas expertos son herramientas fundamentales de apoyo a la toma de decisiones en las diversas áreas del conocimiento, otorgando ventajas competitivas sostenibles a nivel empresarial [1].

2.3.1 Arquitectura Básica de un Sistema Experto

Un sistema experto presenta los siguientes elementos: [1]

- **Base de Conocimientos.** Es la parte del sistema experto que contiene el conocimiento sobre el dominio. Hay que obtener el conocimiento del experto y codificarlo en la base de conocimientos. Una forma clásica de representar el conocimiento en un sistema experto son las reglas. Una regla es una estructura condicional que relaciona lógicamente la información contenida en la parte del antecedente con otra información contenida en la parte del consecuente.

- **Base de Hechos (Memoria de trabajo).** Contiene los hechos sobre un problema que se han descubierto durante una consulta. Durante una consulta con el sistema experto, el usuario introduce la información del problema actual en la base de

hechos. El sistema empareja esta información con el conocimiento disponible en la base de conocimientos para deducir nuevos hechos.

- **Motor de Inferencia.** Es el componente que modela el proceso de razonamiento humano. El motor de inferencia trabaja con la información contenida en la base de conocimientos y la base de hechos para deducir nuevos hechos. Contrasta los hechos particulares de la base de hechos con el conocimiento contenido en la base de conocimientos para obtener conclusiones acerca del problema.

- **Subsistema de Explicación.** Usando el módulo del subsistema de explicación, un sistema experto puede proporcionar una explicación al usuario de por qué está haciendo una pregunta y cómo ha llegado a una conclusión. Este módulo proporciona beneficios tanto al diseñador del sistema como al usuario. El diseñador puede usarlo para detectar errores y el usuario se beneficia de la transparencia del sistema.

- **Subsistema de Adquisición del Conocimiento.** Módulo que permite el mantenimiento de la base del conocimiento. Este módulo es usado por el humano experto y debe ser lo más sencillo posible.

- **Interfaz de Usuario.** La interacción entre un sistema experto y un usuario se realiza en lenguaje natural. También es altamente interactiva y sigue el patrón de la conversación entre seres humanos. Para conducir este proceso de manera aceptable para el usuario es especialmente importante el diseño del interfaz de usuario. Un requerimiento básico del interfaz es la habilidad de hacer preguntas. Para obtener información fiable del usuario hay que poner especial cuidado en el diseño de las cuestiones. Esto puede requerir diseñar el interfaz usando menúes o gráficos.

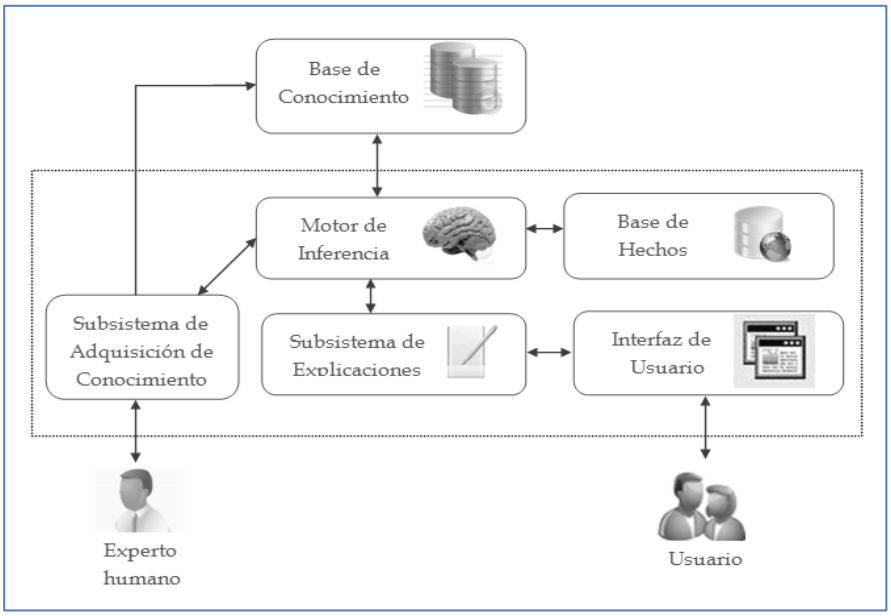


Fig. 2.3. Componentes de un Sistema Experto [1].

2.3.2 Tipos de sistema Experto

Según el sistema deductivo que utilizan para poder llegar a la solución, los Sistemas Expertos se clasifican de la siguiente manera: [1]

- **Basados en reglas.** Los sistemas expertos basados en reglas utilizan para el proceso de inferencia un conjunto de reglas que constituyen la base de conocimiento del experto. Este conjunto de reglas pueden ser activadas a medida que las condiciones son evaluadas positivamente y su utilización implica la creación de nuevos hechos. Este proceso permitirá a partir de unos hechos iniciales desarrollar un proceso deductivo que concluirá el momento en que no quede ninguna otra regla por utilizar. Para realizar este tipo de tratamiento es posible hacerlo de dos maneras diferentes, por un lado realizarlo desde las evidencias hasta los objetivos o por otro lado en orden inverso que sería comenzar desde el objetivo hasta llegar al conjunto de evidencias que lo han provocado.

- **Basados en redes bayesianas.** Este otro tipo de sistema experto basa su funcionamiento como su nombre propio indica en las redes bayesianas. Así pues se trata de un modelo probabilístico que relaciona un conjunto de variables aleatorias mediante un grafo dirigido. El motor de inferencia que utiliza para procesar las evidencias se basa en la teoría de probabilidades y más concretamente con el teorema de Bayes. Este método es especialmente una herramienta extremadamente útil en la estimación de probabilidades ante nuevas evidencias.

- **Basados en casos.** Este último tipo de sistema experto será el que explicaremos con mayor profundidad en los siguientes apartados. Este tipo de sistemas basa su funcionamiento en experiencias anteriormente vividas, ya sea por el propio sistema o bien por la persona experta, y a partir de este conocimiento de vivencias realizar una asociación con estas experiencias para extraer una solución.

2.4 Razonamiento basado en Casos

Rossillea [7], define el Razonamiento basado en Casos (RBC) o Case Based Reasoning (CBR) como “Una técnica de la inteligencia artificial que intenta llegar a la solución de nuevos problemas de forma similar como lo hacen los seres humanos utilizando la experiencia acumulada hasta el momento en acontecimientos similares”.

Según Reyes y Simon [8], “Un nuevo problema se compara con los casos almacenados previamente en la base de casos (Memoria de Casos) y se recuperan uno o varios casos. Posteriormente se utiliza y evalúa una solución sugerida, por los casos que han sido seleccionados con anterioridad, para tratar de aplicarlos al problema actual”.

Un “CASO” es la definición completa, clara y precisa de las características de un problema particular que lo distinguen de entre otros problemas, y las acciones que se deben tomar para su corrección. Un caso debe contar con: Título, Descripción de problema, Criterios de evaluación (preguntas a contestar) y Acciones de solución [2].

2.4.1 Estructura de un sistema CBR

Cuenta con la siguiente estructura: [2]

- **Base de conocimiento de casos.** Es una base de datos de casos históricos estructurada, que captura problemas reales y sus soluciones. Se diseña para almacenar conocimiento y experiencia en el problema. Consta de teorías, principios y clasificaciones del problema, junto con las heurísticas y los juicios asociados a cada caso concreto.

- **Librería de índices.** Formada por un conjunto de índices para buscar y recuperar casos similares al problema actual. El mecanismo de indexado determina los casos que serán seleccionados, mientras que el proceso de recuperación asegura que el caso más relevante es seleccionado para un análisis posterior.

- **Medidas de relevancia.** Son un conjunto predeterminado de características del problema, generales o específicas, que el sistema usa para asegurar la relación entre el caso actual y los históricos. Con estas medidas el sistema puede seleccionar y clasificar los casos más relevantes.

- **Módulo de explicación.** Es el que permite justificar y explicar el análisis completo del problema y las soluciones propuestas, así como la semejanza o diferencia entre dicha solución y las de los casos históricos.

2.4.2 Ciclo CBR

Aamod y Plaza [9], describen CBR como un proceso que consta de cuatro pasos, que engloba un ciclo de razonamiento continuo, estos pasos son los siguientes:

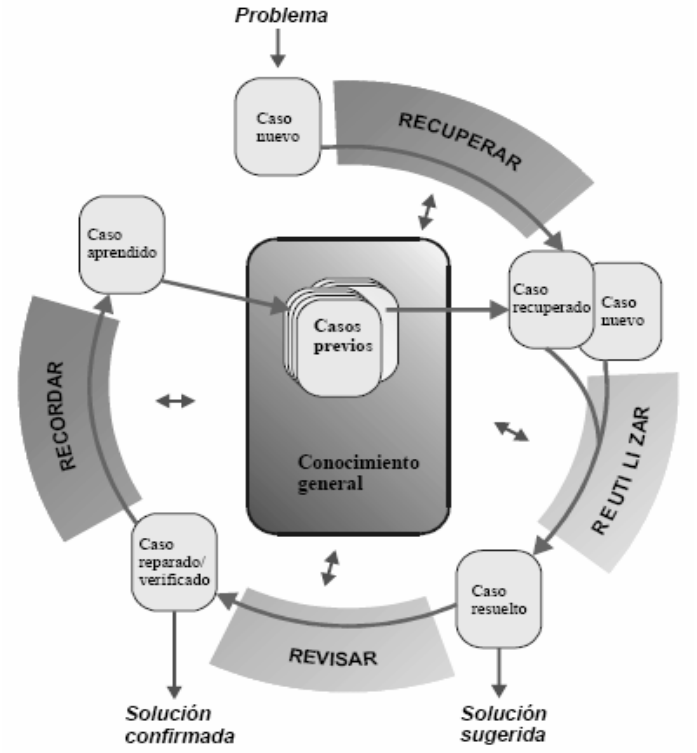


Fig. 2.4. Ciclo del Razonamiento Basado en Casos [9].

1. **Recuperar:** Es la selección, en la base de conocimiento, de aquellos casos cuya descripción se ajusta más a la información presentada en el nuevo caso. Las técnicas más utilizadas son: vecinos más cercanos, recuperación inductiva e Indexado guiado por el conocimiento, las cuales se detallaran en los apartados posteriores.

2. **Reutilizar:** Una vez recuperados los casos similares, se debe tener en cuenta las diferencias entre el caso pasado y el actual, también qué parte o partes del caso recuperado pueden ser transferidas al nuevo caso. En algunos casos la tarea de reutilización se reduce a copiar la solución pasada al nuevo caso, pero en otros casos esta solución no puede ser aplicada directamente y tiene que ser adaptada.

3. **Revisar:** En este paso, la solución generada en la tarea de reutilización se evalúa, y si el resultado es satisfactorio, el nuevo caso y la nueva solución para el caso se almacenan.

4. **Retener:** Este paso consiste en registrar, en la base de conocimiento, la información derivada del nuevo caso, ya sea como un caso nuevo o un caso mejorado, para su posterior reutilización.

2.4.3 Jerarquía de tareas

La visión orientada a procesos del ciclo de vida CBR nos da una buena idea de la secuencia de pasos que se siguen, pero para ver realmente el mecanismo que subyace es necesario tomar una visión orientada a tareas, donde cada paso, o subproceso, se ve como una tarea que el razonador CBR tiene que conseguir. Las tareas se agrupan por las metas del sistema, y una tarea particular se realiza aplicando uno ó varios métodos.

En la figura 2.5, las tareas se muestran con los nombres de los nodos en negrita, mientras que los métodos están en cursiva. Los enlaces entre nodos de tarea (líneas continuas) representan varias descomposiciones de la tarea implicada.

Por ejemplo, la tarea de más alto nivel que es la resolución del problema y aprendizaje de la experiencia se descompone en cuatro tareas, que se corresponden con los cuatro procesos de la figura 2.4, recuperación, reutilización, revisión y almacenamiento. Es necesario llevar a cabo las cuatro tareas para poder alcanzar el objetivo de más alto nivel. A su vez, la tarea de recuperación se descompone en las tareas de identificar, buscar, emparejar y seleccionar. Y así sucesivamente.

Los métodos de cada tarea (líneas discontinuas) indican diferentes formas de llevar a cabo la tarea. Un método especifica un algoritmo que identifica y controla la ejecución de la subtarea particular, utilizando el conocimiento e información necesarios. Por ejemplo, para la tarea de más alto nivel (recordemos que es la resolución de un problema y el aprendizaje de la experiencia), el método para llevarla a cabo es el razonamiento basado en casos.

La figura es completa en cuanto a las particiones que realiza, es decir, el grado de descomposición de las tareas en subtareas pretende ser suficiente para llevar a cabo la misma. Por supuesto, la figura no muestra ninguna estructura de control en las subtareas,

si bien, se da a entender cierto secuenciamiento en cuanto que se han colocado antes (referente a posición en la página) las subtareas que aquellas que las siguen. Sin embargo, la figura no es completa en cuanto a los métodos, es decir, uno de los métodos que se indican puede ser suficiente para resolver la tarea, o puede que se deban combinar varios de ellos, o en el caso más extremo se deban utilizar otros métodos aquí no indicados [9].

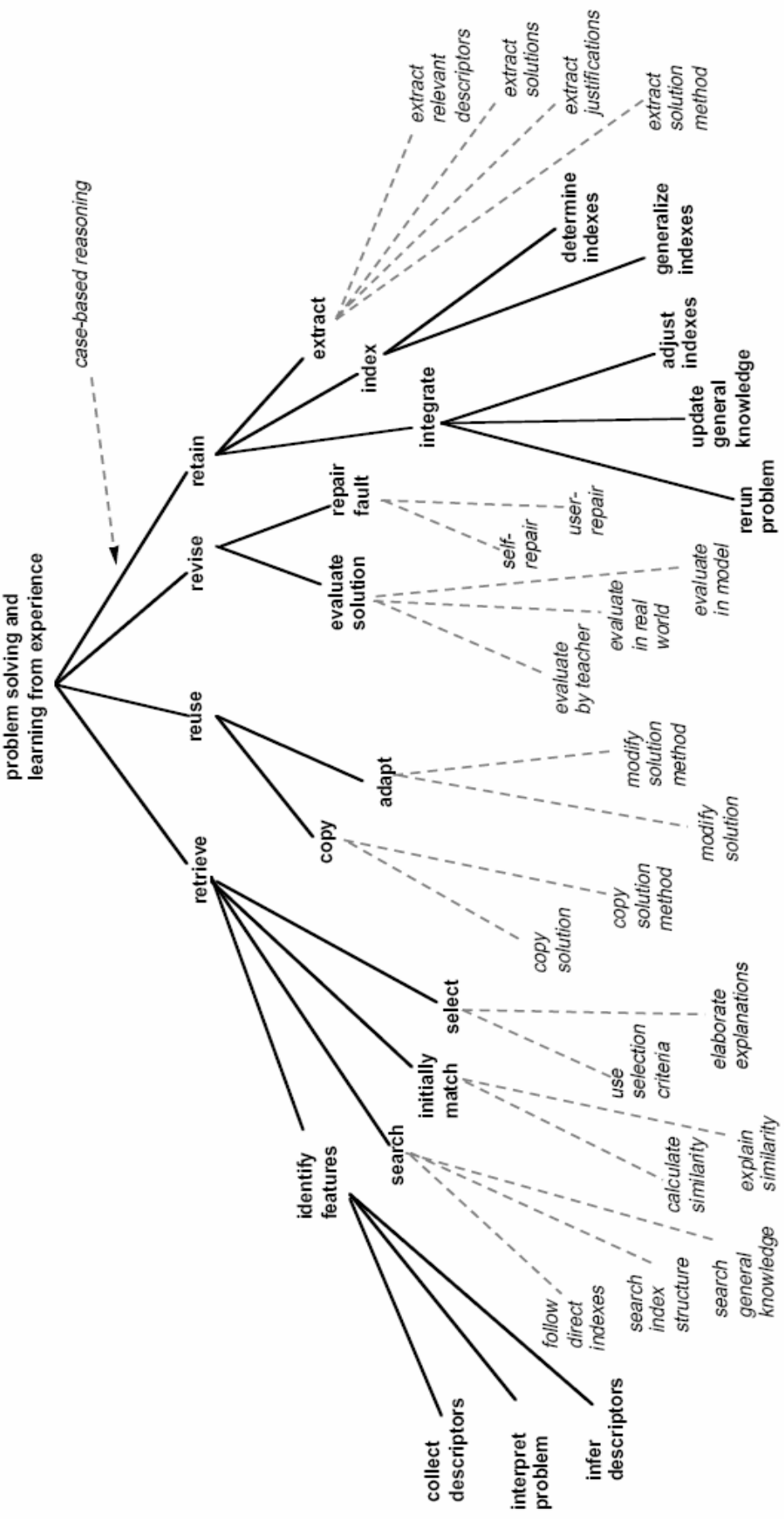


Fig. 2.5. Jerarquía de Tareas

2.4.4 Ventajas e inconvenientes

Entre las principales ventajas de estos sistemas, tenemos: [2]

- Provee gran flexibilidad en el modelado del conocimiento. En contraste con los sistemas basados en modelos, donde muchos problemas no se resuelven por considerarlos fuera de su ámbito o no se entiende el problema a poco que los datos estén incompletos, los sistemas CBR usan las experiencias pasadas como dominio de conocimiento y dan una solución razonable, previa adaptación, de este tipo de problemas.

- Reducen la tarea de adquisición del conocimiento. Eliminando la tarea de extraer de un modelo o de un conjunto de reglas, como en los sistemas basados en modelos/reglas, la tarea de adquisición de conocimiento basado en un razonador basado en casos consiste en una representación de experiencias/casos almacenados junto a su representación y solución.

- Aprende con el tiempo. Esto no es obvio en la mayoría de los sistemas de resolución de problemas. Sin embargo, como ya se ha citado, a medida que los razonadores basados en casos son más usados, encuentran más situaciones de problemas y crean más soluciones. Tras las evaluaciones, determinando un nivel de éxito en las soluciones, los casos se añaden al sistema y el sistema tendrá más variedad de situaciones y más grados de refinamiento y éxito.

- Provee un medio de justificación. El razonamiento basado en casos puede dar un caso previo y una solución (con éxito) de forma que puede utilizarse para convencer al usuario o para justificar una solución propuesta al problema actual. Si el usuario deseara una medida de calidad de la solución, el sistema podría cuantificar cuánto éxito tuvo el caso pasado y que grado de similitud hay con el caso actual y el pasado.

- Evita la repetición de errores del pasado. En los sistemas que guardan tanto los fallos como los éxitos, así como las causas de los fallos, se utiliza la información acerca de qué causó el fallo pasado para predecir posibles fallos futuros. El sistema puede incluso alertar al razonado para que tome las acciones necesarias para no repetir errores.

- Permite al razonador proponer soluciones en dominios que no son del todo entendidos por el razonador. En situaciones donde existe poco conocimiento para construir un modelo causal del dominio o para derivar un conjunto de heurísticas, un razonador basado en casos sí puede desarrollar un pequeño conjunto de casos que le permitan funcionar.

- Hay dominios que son imposibles de entender completamente porque dependen del comportamiento humano impredecible (la economía por ejemplo). Otros, porque simplemente no se ha alcanzado el nivel adecuado para su comprensión. Sin embargo, el razonamiento basado en casos permite tomar ciertas premisas y predicciones basándonos en lo que funcionó en el pasado.

- Permite hacer predicciones del posible éxito de una solución propuesta. Cuando la información se almacena teniendo en cuenta el nivel de éxito de las soluciones previas, el razonador basado en casos puede ser capaz de predecir el éxito de una solución propuesta para el problema actual.

- Obviamente, el razonador tendrá en cuenta no sólo esos niveles de éxito almacenados sino las diferencias entre el caso ó casos recuperados y la situación actual.

- Se propone soluciones al problema rápidamente. En dominios que requieren un gran procesamiento para crear una solución de la nada (por ejemplo, recorriendo un grafo), el razonador puede reducir ese tiempo considerablemente tomando una solución temprana y modificándola adecuadamente.

- Se puede utilizar para muchos propósitos, como crear planes, diagnósticos, argumentación de puntos de vista, etc. de formas muy distintas, dependiendo básicamente de los métodos de recuperación y de adaptación que implementen.

- Es un reflejo del razonamiento humano, lo que es una gran ventaja a la hora de entender el funcionamiento del sistema, así como la justificación de la solución propuesta por un sistema basado en casos.

Las principales desventajas que presentan son: [2]

- Puede haber una tendencia a usar los casos previos ciegamente, confiando en la experiencia previa sin validarla con respecto a la nueva situación.

- Los casos previos pueden predisponer demasiado al razonador a la hora de resolver el nuevo problema

- Es común que normalmente no se disponga del conjunto de casos más apropiado para el tratamiento de un problema concreto.

- Confiar en experiencias previas sin validar puede generar soluciones y evaluaciones ineficientes o incorrectas. La recuperación de casos inapropiados puede costar un tiempo considerable.

2.4.5 Problemática CBR

Cada una de las diferentes fases que se definen para un sistema CBR tiene su problemática propia [2].

2.4.5.1 Indexación

El problema de la indexación viene a responder al problema de accesibilidad de los datos. Hace referencia a la selección de los atributos más importantes, de estructura y organización de los casos y los índices de recuperación de los casos más importantes de forma correcta.

2.4.5.2 Recuperación

Se debe elegir entre ejecutar una búsqueda basada en similitudes sintácticas o en similitudes semánticas (esta última solo si se requieren explicaciones sobre la solución inferida). Por lo tanto es importante saber la orientación que debe tener el sistema, sus metas y objetivos, de forma que se seleccionen los métodos apropiados sobre las características mencionadas en la indexación. Es importante además seleccionar adecuadamente los métodos de medida de similitud entre los casos recuperados y la situación actual.

2.4.5.3 Reutilización

La reutilización de casos es el proceso de transformar una solución recuperada en una solución apropiada para el problema actual. Se ha llegado a argumentar que la reutilización es el paso más importante del razonamiento basado en casos, ya que incorpora inteligencia a lo que, de no ser así, sería un mero proceso de reconocimiento de patrones.

2.4.5.4 Aprendizaje

En cada dominio y aplicación es necesario decidir cuál, cuándo y cómo se debe almacenar una situación como un caso resuelto (satisfactoriamente o no) que enseña algo nuevo al sistema. Es habitual el disponer de una base de casos preestablecida desde el inicio del sistema, siendo la fase de aprendizaje necesaria en algunos casos y en otros se reduce a realizar simples instrucciones de almacenamiento en una base de casos. Existen diversas teorías de cómo establecer y construir estas bases de casos intentando mantener la consistencia o mantener la verdad de los casos en ella.

2.5 Enterprise Resource Planing (ERP)

2.5.1 Definición

Ramírez Correa [3], define ERP como “Una extensa solución comercial de software empaquetado compuesto de varios módulos configurables que integran, firmemente y en un solo sistema las actividades empresariales nucleares - finanzas, recursos humanos, manufactura, cadena del abastecimiento, gestión de clientes - a través de la automatización de flujos de información y el uso de una base de datos compartida. Incorporando en este proceso de integración las mejores prácticas para facilitar la rápida toma de decisiones, las reducciones de costos y el mayor control directivo, y logrando con ello el uso eficiente y eficaz de los recursos empresariales”.

2.5.2 Arquitectura

La arquitectura de un ERP, puede ser vista desde dos perspectivas, la primera asociada a la funcionalidad del sistema y la segunda, a las características técnicas de ellos.

2.5.2.1 Perspectiva Funcional

Según Rashid [10], desde una perspectiva funcional, debemos indicar los que sistemas ERP están diseñados en forma modular, es decir, como piezas de un gran mecano.

Cada uno de estos módulos o aplicaciones - conjunto de programas computacionales - tiene una función específica. Cada organización determina que partes de este mecano necesita utilizar al momento de implantar el paquete de software [10].

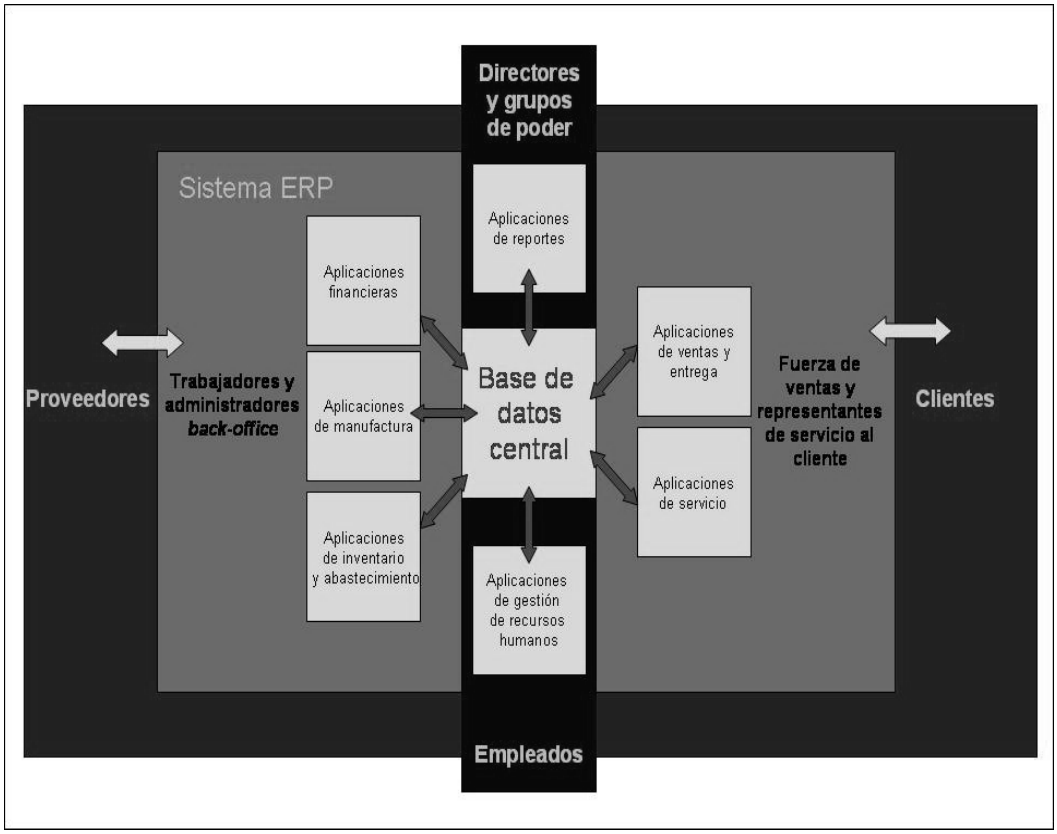


Fig. 2.6. Arquitectura de un Sistema ERP [10].

El concepto de modularidad de un sistema ERP se puede ilustrar siguiendo la figura 2.6., propuesta por Davenport [10], en ella se puede apreciar en la parte central del sistema ERP una base de datos que tanto capta la información que proviene de distintas aplicaciones, como a su vez entrega desde sus repositorios la información que estas aplicaciones necesitan para apoyar a las diversas funciones de la empresa. En relación a los módulos o aplicaciones, podemos indicar que, primero y más en cerca de los proveedores, las aplicaciones financieras, de manufactura, de inventario y abastecimiento sirven a los trabajadores y administradores de tipo back-office. Más cercana a los clientes un segundo grupo de aplicaciones de ventas, entrega y servicio apoyan tanto a las fuerzas de venta como a los representantes del servicio al cliente. Adicionalmente, los dos grupos de aplicaciones nombradas se integran con las aplicaciones de gestión de recursos humanos y las aplicaciones de reportes a directivos y cargos gerenciales [11].

Las funciones de los sistemas ERP se pueden clasificar en cuatro grandes grupos, dependiendo del proceso de negocios que apoyen: procesos de manufactura, procesos financieros y contables, procesos de ventas y marketing, y procesos de recursos humanos.

2.5.2.2 Perspectiva Técnica

Desde una perspectiva técnica, los sistemas ERP actuales están diseñados y construidos utilizando dos elementos técnicos, una arquitectura cliente/servidor para su operación, y una base de datos relacional que organiza todos los datos necesarios para soportar las funcionalidades antes comentadas [3].

La arquitectura cliente/servidor es una configuración computacional descentralizada que se basa en que existe un computador llamado servidor que entrega servicios a un conjunto de computadores llamados clientes [3].

Las Bases de Datos Relaciónales (BDR) son un estándar en el actual desarrollo de sistemas computacionales para la empresa y su denominación deriva del uso de un modelo específico para organizar los datos. Una base de datos se puede definir como una colección de datos organizada para dar servicio eficiente a muchas aplicaciones al centralizar los datos y minimizar aquellos que son redundantes [3].

2.5.3 Ventajas y desventajas

Los sistemas ERP, presentan las siguientes ventajas y desventajas [4]:

2.5.3.1 Ventajas:

- Sistema totalmente integrado
- La capacidad para racionalizar los diferentes procesos y flujos de trabajo
- La posibilidad de compartir fácilmente los datos a través de varios departamentos de una organización
- Mejora de la eficiencia y los niveles de productividad
- Mejor seguimiento y previsión
- Costes más bajos
- Mejora el servicio al cliente

2.5.3.2 Desventajas:

- Limitada personalización (Dependiendo del producto)
- La necesidad de procesos Reingeniería
- Costos altos
- Pueden ser demasiado rígidos o específicos.

2.5.4 Implantación

2.5.4.1 Medidas de éxito de la implantación

Según Markus [12], el éxito de la implantación de un sistema ERP depende, primero, del punto de vista del cual se mida este resultado y, segundo, del momento en cual se mida.

Con relación al punto de vista de evaluación, es posible suponer que los consultores que instalan el software tendrán una visión claramente distinta del éxito que las personas que trabajan dentro de la organización en la cual se implantó el sistema, quizás los primeros enfatizaran un éxito en función de cumplir con los plazos y presupuestos establecidos en la planificación, y para los segundos este éxito estará asociado a mejoras de negocio, como pueden ser la reducción del inventario o el aumento de la velocidad en la toma de decisiones [12]. Se estiman que existen al menos cinco dimensiones en las cuales se valora este éxito:

- Éxito visto en términos técnicos.
- Éxito visto en términos del negocio, ya sean económicos, financieros o estratégicos.
- Éxito visto en términos del funcionamiento sin problemas de las operaciones de la organización.
- Éxito según la visión de los directivos y los empleados de la organización que adopta el ERP.
- Éxito según la visión de clientes, proveedores, e inversionistas de la organización que adopta el ERP.

2.5.4.2 Metodologías de implantación

A continuación se describirán tres de las metodologías estándares para implantación de ERPs, cabe resaltar que existen muchas otras metodologías, para productos específicos e incluso metodologías propias de las organizaciones que se dedican a la implantación de estos sistemas. En este punto abarcará principalmente las siguientes propuestas:

1) El modelo “Ciclo de la experiencia ERP” de Markus y Tanis [13].
2) El “modelo de proyecto por fases (PPM)” de Parr y Shanks [13].
3) La metodología “Ciclo de vida de un sistema ERP” de Ahituv [14].

2.5.4.2.1 Ciclo de la experiencia ERP

Markus y Tanis [13], desarrollaron este modelo de la implantación de sistemas ERP, el cual propone un ciclo de cuatro fases, que se muestran en la Figura 2.7. :

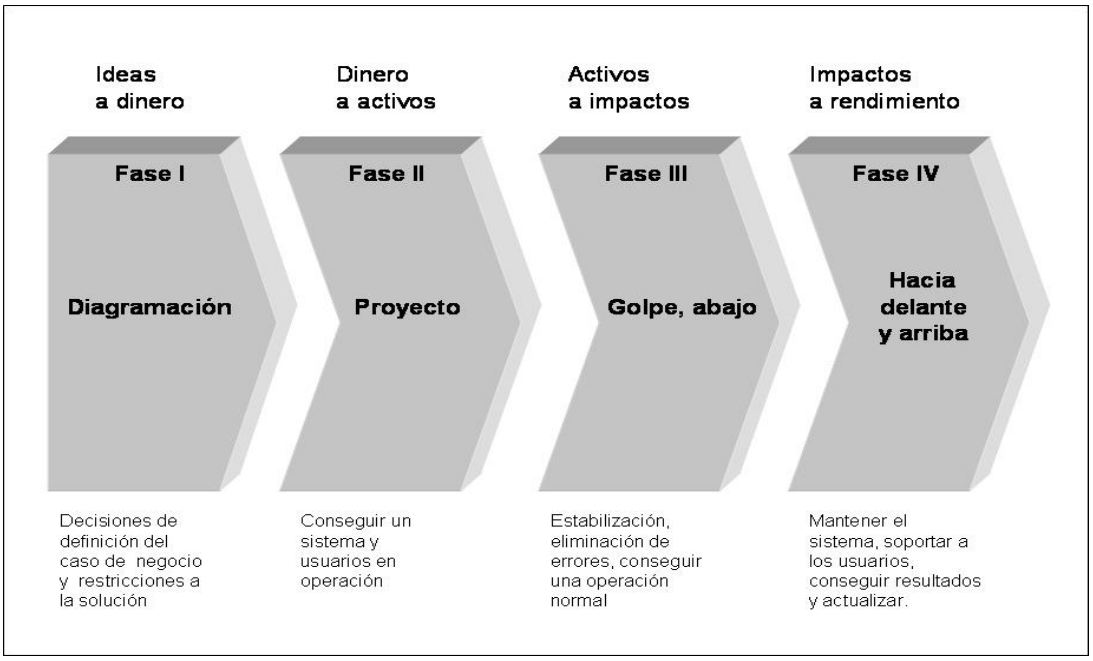


Fig. 2.7. Ciclo de la Experiencia ERP [13].

- La fase llamada "diagramación" (chartering) comprende las decisiones que conducen al financiamiento de un sistema ERP en la organización, esta fase debe transformar las ideas en dinero. Entre los actores claves de esta fase se incluyen los proveedores, consultores, directivos de la organización, y especialistas en tecnologías de información. Las actividades claves de esta fase incluyen la construcción de un caso de negocio de los sistemas empresariales, la selección de un paquete de software (aunque esta decisión se puede diferir hasta la fase del proyecto), la identificación de un encargado de proyecto, y la aprobación de un presupuesto y una planificación temporal.

- La fase llamada "proyecto" comprende las actividades previstas para conseguir que el sistema esté en servicio en una o más unidades de organización, esta fase debe transformar el dinero en activos. Los actores claves en esta fase incluyen al encargado de proyecto, los miembros del equipo de proyecto (a menudo existen miembros no técnicos de distintas unidades de negocio y áreas funcionales), los especialistas internos en tecnologías de información, los proveedores, y los consultores. Las actividades claves incluyen la configuración del software, la integración del sistema, la prueba, la conversión de datos, y el entrenamiento. Los problemas que surgen en esta fase se relacionan con las modificaciones del software, la integración del sistema, las dificultades con el producto y los consultores, y la rotación de personal del proyecto.

- En la fase llamada “golpe, abajo” (shakedown) la organización comienza a sufrir las complicaciones del sistema ERP. Esta fase debe transformar los activos en impactos, y se puede decir que termina cuando se ha alcanzado la “operación normal”. El equipo del proyecto (o los consultores) podrían continuar con su implicación o podrían pasar el control del sistema a los administradores operacionales y usuarios finales, pero siempre apoyarán a resolver las dudas técnicas. Los problemas que surgen en esta fase se asocian a una perspectiva excesivamente funcional en la implantación, el inapropiado corte del alcance del proyecto, un entrenamiento breve del usuario final, pruebas inadecuadas del sistema, la no realización de mejoras en los procesos de negocio previo a la implantación del sistema, y la infravaloración de los problemas de calidad de datos y de las necesidades de creación de reportes.

- La última fase llamada “hacia adelante y arriba” (onward and upward) continúa la operación normal del sistema hasta que éste se substituye por una versión mejorada o un sistema diferente. Esta fase debe transformar los impactos en mejoras del rendimiento, durante ella la organización puede finalmente comprobar las ventajas, si es que hay, de su inversión. Los actores claves en esta etapa incluyen tanto a los administradores operacionales y usuarios finales, como al personal de tecnologías de información de soporte (interno y/o externo). El personal del proveedor y los consultores también pueden estar involucrados, particularmente cuando se consideran mejoras al sistema. Esta fase conlleva problemas relacionados con resultados de negocio desconocidos, resultados de negocio mucho menor a los esperados, la fragilidad del capital humano, y las dificultades de migración del sistema ERP.

2.5.4.2.2 Modelo de proyecto por fases (PPM)

Parr y Shanks [13], presentan una aproximación a la implantación de sistemas ERP llamado modelo de proyecto por fases (Project Phase Model - PPM).

Tal como lo indica la Figura 2.8., PPM propone tres fases importantes en la implantación de un ERP: I) Planeación; II) Proyecto; y III) Mejora [13].

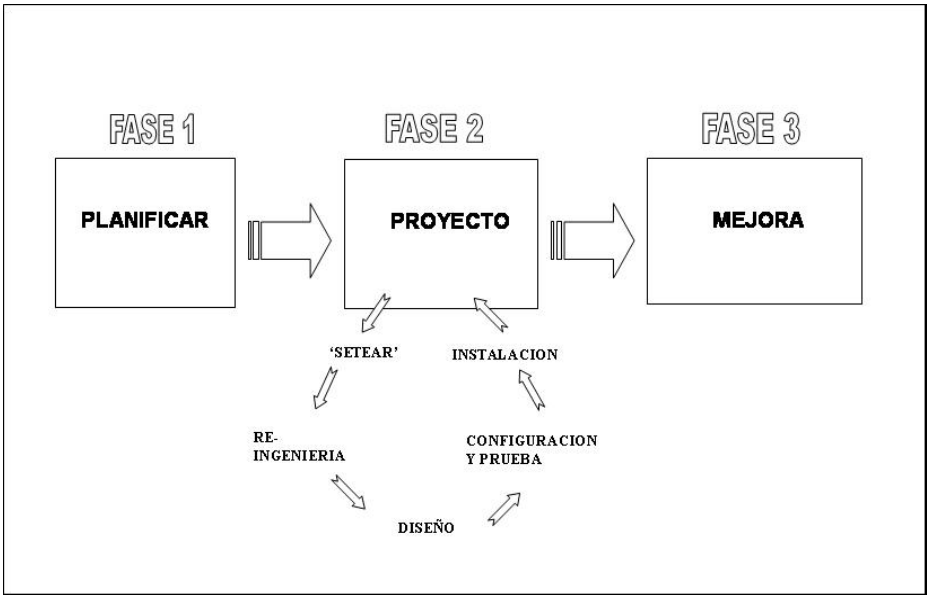


Fig. 2.8. Modelo de Proyecto por fases [13].

- La fase del Planeación incluye la selección de un sistema ERP, el establecer un comité de dirección, la determinación a alto nivel del alcance del proyecto y la aproximación general que se utilizara en la implantación, la selección de un director del equipo de proyecto, y la determinación de los recursos necesarios para llevar a cabo la implantación [13].

- La fase del Proyecto se extiende desde la identificación de los módulos de ERP necesarios para la organización hasta su instalación y puesta en marcha [13].

- La fase de Mejora puede extenderse por varios años e incluye las etapas de reparación, extensión y transformación del sistema. De hecho, la gestión y el soporte al ERP son preocupaciones continuas de las organizaciones usuarias de estos sistemas [13].

El foco del modelo PPM está en la fase del Proyecto, esta fase se divide en cinco subfases: 1) 'Setear' (disposición inicial); 2) Re-ingeniería; 3) Diseño; 4) Configuración y prueba; e 5) instalación.

- 'Setear' (disposición inicial): En la subfase de 'Setear' el equipo(s) del proyecto es seleccionado y estructurado con una mezcla apropiada de conocimiento técnico y del negocio, se establecen tanto el equipo(s) de integración como los procesos de reporte, como así mismo, se desarrollan y/o reafirman los principios que guiarán el proyecto [13].

- Re-ingeniería: La subfase del re-ingeniería implica, por una parte, el análisis de los procesos del negocio actuales para determinar el nivel de ingeniería de proceso del negocio requerida, y por otra, la instalación del ERP, el desarrollo de un mapa que relacione los

procesos del negocio a las funciones del sistema ERP, y el entrenamiento del equipo(s) de proyecto [13].

- Diseño: La subfase del diseño implica primero un diseño de alto nivel y luego un diseño detallado conforme a la aceptación del usuario. Este diseño es acompañado con una interactiva prototipación y una comunicación constante con los usuarios [13].

- Configuración y prueba: Las principales actividades del subfase de configuración y prueba son desarrollar una configuración completa, poblar el ambiente de prueba con datos reales, construir y probar las interfaces de datos, escribir y probar los informes y, por último, realizar las pruebas de sistema y de usuario [13].

- Instalación. Finalmente, la subfase instalación incluye la construcción de redes informáticas, la instalación de las computadoras de escritorio, y la gestión del entrenamiento y el soporte a los usuarios [13].

2.5.4.2.3 Ciclo de vida del sistema ERP

Ahituv [14], propone una metodología propia para la implantación de sistemas ERP que llaman Ciclo de vida del sistema ERP, descrito en la Figura 2.9.

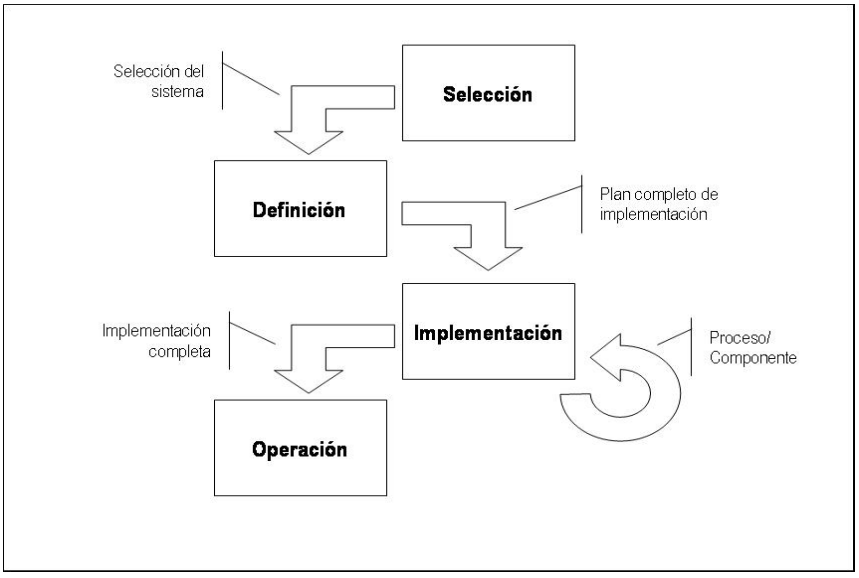


Fig. 2.9. Ciclo de la Vida del Sistema ERP [14].

- **FASE I:** Selección. El objetivo de esta fase es identificar el paquete ERP apropiado para la organización, así como la infraestructura tecnológica necesaria para la operación de este software. Cuando la organización opta por instalar componentes de distintos paquetes ERP, el objetivo de esta fase es determinar cada uno de estos componentes. Si la organización requiere recursos humanos externos para la implantación del paquete, tales como consultores o

expertos, en esta fase se deberán identificar esas empresas proveedoras. Esta fase tiene nueve actividades:

1. Definición de objetivos del proyecto.

2. Recolección de información acerca de los sistemas ERP y sus proveedores.

3. Recolección de información acerca de las firmas de consultaría.

4. Análisis de necesidades de la organización.

5. Investigación de alternativas propuestas por los proveedores, luego de su recepción.

6. Investigación de alternativas propuestas por los consultores, actividad que se realiza en paralelo con la actividad 5.

7. Recolección de información sobre la infraestructura tecnológica necesaria para la operación del ERP, esto a partir de propuestas de los proveedores.

8. Estudio de factibilidad técnica, económica y organizacional de las diversas alternativas de paquetes ERP ofertadas a la organización.

9. Negociación y firma del contrato.

- **FASE II:** Definición. Esta es la fase más corta de las cuatro e incluye todas las actividades preparatorias de las siguientes fases. Esta fase comprende los siguientes pasos:

  1. Definición del alcance del proyecto, es decir, cuales son los límites de los cambios deseados por la inclusión del sistema dentro de la organización.

  2. Determinar el equipo de implantación y establecer una planificación temporal para las actividades de implantación.

  3. Entrenamiento del equipo de implantación en el paquete ERP.

  4. Implantación inicial del paquete ERP en un ambiente de entrenamiento, en paralelo a la actividad anterior.

- **FASE III:** Implantación. El objetivo de esta fase es ligar al sistema ERP a los procesos de la organización hasta el punto que el sistema esté operativo. Esta fase se realiza en forma iterativa, adicionando ya sea procesos y/o capas organizacionales. Esta forma iterativa de implantación, por una parte, reduce el riesgo del proyecto por una temprana detección de problemas de implantación, y por otro lado,

posibilita el mejoramiento de la implantación, entrenando al equipo y reevaluando los siguientes pasos del ciclo de proyecto. Esta característica de interacción es única y diferencia esta metodología del ciclo de vida de sistemas. Esta fase de implantación comprende nueve pasos:

1. Análisis de la diferencia (GAP) entre la definición de procesos del paquete ERP y los procesos de la organización.

2. Reingeniería de procesos de negocio a los flujos de trabajo y procesos de la organización, este paso se deberá realizar antes o en paralelo al paso anterior.

3. Identificación de soluciones complementarias al paquete ERP, como pueden ser la compra de otro software que se integre con el ERP, o la adicción de procesos de trabajo manual.

4. Construcción de un prototipo del sistema ERP, luego de todas las iteraciones este prototipo terminará siendo el sistema utilizable.

5. Conversión de los datos, actividad que se realiza en paralelo con el paso 4 y que puede ser manual o usando software de conversión.

6. Definición de nuevos procedimientos de trabajo o actualización de los existentes.

7. Total implementación del sistema en una unidad, luego que el prototipo está terminado y los datos convertidos.

8. Entrenamiento de usuarios.

9. Pruebas de aceptación del sistema ERP con datos reales.

- **FASE IV:** Operación. Esta es la fase más larga del ciclo de vida del ERP y puede durar varios años. Loa pasos de esta fase son cinco:

  1. Establecer un centro de soporte para asistir a los usuarios en la operación del ERP.

  2. Ejecución de cambios y mejoras, como respuesta a la dinámica propia de la organización, como también de los cambios tecnológicos, estratégicos y ambientales.

  3. Actualización del sistema ERP por nuevas versiones proporcionadas por el proveedor.

  4. Auditoria periódica del sistema para determinar si satisface las necesidades de los usuarios.

5. Término del sistema, cuando el ERP ya no satisfaga las necesidades de la organización.

2.6 jColibri

Para un mejor entendimiento de esta herramienta, previamente se darán a conocer las siguientes definiciones:

2.6.1 Framework

De acuerdo con Gamma [15], "Un Framework es un conjunto de clases cooperantes que componen un diseño reusable para un tipo específico de software. El Framework determina la arquitectura de su aplicación, define su estructura global, su partición en clases y objetos, las principales responsabilidades, cómo los objetos y las clases cooperan entre sí, y el hilo de control principal".

2.6.2 Ontología

Gruber [16], define ontología como "Una especificación formal y explícita de una conceptualización compartida, por lo tanto, la ontología hace referencia a una representación explícita y expresada en un lenguaje formal del modelo conceptual de un cierto dominio de conocimiento. Esta característica de compartición es lo que permite reutilizar el vocabulario semántico de una ontología en la construcción de diferentes bases de conocimiento e incluso de diferentes sistemas inteligentes, siempre que estos trabajen con la misma perspectiva sobre un mismo dominio de conocimiento".

Las ontologías pueden ser utilizadas para distintas tareas dependiendo del tipo de aplicación donde se integren, según Van Heijst [65] y Chandrasekaran [67].

A continuación se detalla una breve clasificación, que en ningún modo es excluyente ya que una misma ontología puede enmarcarse dentro de varios de los siguientes tipos:

- Las ontologías de representación de conocimiento, según Van Heijst [65], capturan las primitivas de representación utilizadas para formalizar conocimiento en algún paradigma de representación. Estas ontologías proporcionan un marco de representación neutral respecto a las entidades concretas del mundo o del dominio que describen.

- Las ontologías de conocimiento general o de sentido común capturan vocabulario relativo a cosas generales como eventos, tiempo, causalidad, espacio, comportamiento, etc.

- Las ontologías de nivel superior (Top-Level) proporcionan elementos y términos con alto nivel de abstracción, es decir, términos generales bajo los cuales se suelen colocar los términos más específicos de otras ontologías.

- Las ontologías del dominio, según Van Heijst [65] y Riichiro [66], proporcionan vocabulario sobre los conceptos de un dominio y las relaciones entre ellos, las actividades que se desarrollan y los principios elementales que rigen el comportamiento en ese domino.

- Las ontologías lingüísticas proporcionan información sobre los elementos de un idioma. La más importante es WordNet, según Miller [68], y contiene una base de datos léxica para el idioma inglés cuya información se organiza en unidades llamadas synsets o conjuntos de términos sinónimos.

- Las ontologías de tareas, según Riichiro [66], proporcionan un vocabulario sistemático de los términos utilizado para resolver problemas asociados con tareas compartidas por distintos dominios. Relacionadas con éstas se encuentran las llamadas ontologías de tareas-dominios que son ontologías de tareas que sólo se pueden reutilizar en un cierto dominio y no en otros, debido a que la terminología asociada a la tarea está particularizada a un dominio concreto.

- Las ontologías de métodos de resolución de problemas proporcionan definiciones de los términos, conceptos y relaciones que se utilizan para especificar un proceso de razonamiento llevado a cabo para lograr una tarea, según Chandrasekaran [67]. Son independientes del dominio y sus elementos se describen en términos neutrales respecto a una aplicación a un dominio concreto.

- Las ontologías de aplicación, según Van Heijst [65], contienen el conocimiento y las definiciones necesarias para una cierta aplicación.

El conocimiento contenido en las ontologías se formaliza por medio de cinco tipos de componentes: clases, relaciones, funciones, axiomas e instancias, según Gruber [16]. Aunque las clases de una ontología se organizan generalmente de forma taxonómica, no se debe considerar esta taxonomía como una ontología en sí misma. El término concepto o clase se utiliza en un sentido amplio para agrupar conjuntos de individuos (también denominados instancias), que pueden describir a su vez tareas, funciones, acciones, estrategias, procesos, etc.

Las relaciones representan un tipo de interacción entre los conceptos del dominio. Las funciones son un caso especial de relación n-aria en el que el elemento n de la relación está unívocamente determinado por los n-1 anteriores elementos. Los axiomas se usan para modelar sentencias que son siempre ciertas en este dominio y las instancias se usan para representar individuos concretos. Una vez que los componentes de la ontología hayan sido identificados, la ontología puede ser implementada en algún lenguaje. Existen numerosos lenguajes para la representación de ontologías como: CML, Ontolingua, o LOOM. Sin embargo ninguno de ellos ha obtenido la popularidad de OWL. Este es el lenguaje propuesto por el World Wide Web Consortium (W3C) para la representación de ontologías dentro de la Web Semántica y se ha convertido en un estándar completamente implantado.

jColibri es un armazón o Framework orientado a objetos que facilita la construcción de sistemas de razonamiento basado en casos (CBR). Un armazón es una aplicación reutilizable y semi-completa que puede ser especializada para obtener aplicaciones concretas [25].

jColibri ha sido diseñado pensando en conseguir una plataforma de desarrollo de sistemas CBR que sirva de referencia en la comunidad científica. El objetivo es conseguir que el armazón crezca y evolucione gracias a las aportaciones de distintos especialistas [25].

Para conseguir este ambicioso objetivo, la arquitectura de jColibri utiliza técnicas desarrolladas en el campo de la Ingeniería del Software para promover la reutilización del software, y las ideas propuestas por la metodología KADS [20], que básicamente consiste en separar los métodos que obtienen la solución a un problema del modelo del dominio, que describe el conocimiento específico de cada aplicación. jColibri está desarrollado en Java lo que facilita su utilización en distintos entornos, y utiliza tecnologías ampliamente aceptadas como XML (Extensible Markup Language o Lenguaje de Marcas) o JDBC (Java Database Conectivity) [25].

Además, jColibri proporciona una herramienta que permitirá construir aplicaciones CBR concretas (instanciar el armazón) de forma visual y guiada. Aunque actualmente la herramienta no llega a generar una aplicación completa, sirve para crear el esqueleto de la aplicación. Este tipo de herramientas alivian la curva de aprendizaje inicial a la que se suele enfrentar cualquiera que desee aprender a utilizar un armazón [25].

2.6.3 Principales Características

2.6.3.1 Evolución Histórica

jColibri es la evolución de un sistema anterior denominado COLIBRI [21]. Este sistema se centraba en sistemas CBR con conocimiento intensivo (K-I CBR). Estos sistemas se caracterizan por utilizar conocimiento adicional acerca del dominio sobre el que operan lo que, en teoría, debería mejorar su eficiencia. El sistema COLIBRI utilizaba una ontología de conceptos comunes en sistemas CBR denominada CBROnto [22]. Esta ontología definía un vocabulario común sobre el que se había desarrollado una biblioteca de métodos de resolución de problemas (Problem Solving Methods o PSM). Esta biblioteca permitía resolver tareas comunes a distintos sistemas CBR sin necesidad de utilizar conocimiento específico del dominio, lo que fomentaba su reutilización. COLIBRI se desarrolló en LISP y utilizaba LOOM [23] como tecnología para representación del conocimiento. Este sistema validó la viabilidad de la propuesta, pero era poco utilizable fuera del grupo de investigación donde fue desarrollado. jColibri hereda muchas de las ideas originales de este sistema, e intenta convertirse en una herramienta utilizable por la comunidad científica [25].

2.6.3.2 Sistemas CBR con conocimiento Intensivo

Cualquier sistema que utilice de forma intensiva conocimiento se encuentra con el problema que supone adquirirlo. jColibri intenta paliar este problema mediante la reutilización del conocimiento, y para hacerlo facilita la utilización de ontologías en las aplicaciones CBR [25].

Existen aplicaciones especializadas en el uso de ontologías para clasificar, inferir, encontrar inconsistencias entre conceptos y relaciones, etc. jCOLIBRI proporciona los mecanismos necesarios para comunicar la aplicación CBR desarrollada con estos

sistemas externos, en especial con RACER [24] que es un motor basado en lógicas descriptivas [25].

Una característica importante de jCOLIBRI es que permite utilizar ontologías para indexar los casos. De esta forma los casos son instancias dentro de una jerarquía conceptual que soporta sofisticados mecanismos de recuperación, adaptación y aprendizaje a través de aplicaciones externas como RACER [25].

2.6.3.3 Tareas y Métodos

jColibri ha sido construido en torno a la idea básica de separar tareas (tasks) y métodos (methods). Las tareas indican objetivos que el sistema debe alcanzar y guían la ejecución de la aplicación. Los métodos indican como resolver las tareas [25].

Por ejemplo, una idea bastante aceptada es que el ciclo principal de CBR puede descomponerse en cuatro tareas: recuperar los casos mas similares (retrieve), reutilizarlos para resolver el problema (reuse), revisar la solución propuesta (revise) y aprender de la experiencia (retein) [25].

Siguiendo esta idea, jColibri modela el ciclo CBR mediante la tarea CBR Task y propone un método asociado CBR Method que la descompone en 4 subtareas: CBR Retrieve Task, CBR Reuse Task, CBR Revise Task y CBR Retein Task. A su vez para resolver estas tareas habrá que seleccionar los métodos correspondientes [25].

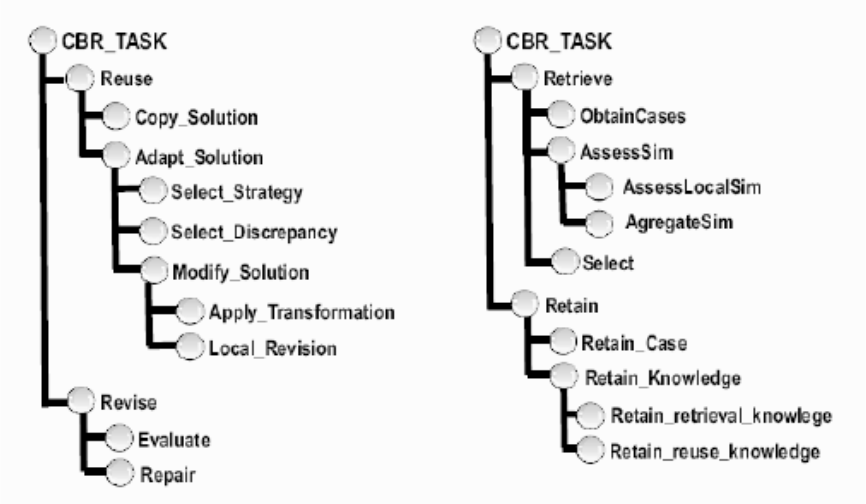


Fig. 2.10. Representación de CBR Method [25].

Actualmente los métodos pueden ser de dos tipos: métodos de descomposición, que descomponen una tarea en varias subtareas (como CBR Method visto anteriormente en la Figura 2.10), o métodos de resolución, que resuelven directamente una tarea mediante una clase java.

Por lo tanto una aplicación CBR puede representarse como una estructura en forma de árbol con las tareas que deben resolverse. Ejecutar una aplicación consiste en resolver dichas tareas ejecutando los métodos correspondientes [25].

Tanto las tareas como los métodos se describen utilizando términos de una ontología sobre CBR independiente del dominio (CBROnto). Esta ontología define conceptos que se utilizan en el ámbito del razonamiento basado en casos como por ejemplo: tarea, método, caso, base de casos, función de similitud, consulta..., definiendo un vocabulario común sobre el que se puedan construir métodos genéricos que resuelvan problemas sin necesidad de conocimiento específico del dominio [25].

En la implementación de jColibri los conceptos de CBROnto se representan como clases abstractas o interfaces java [25].

Esta arquitectura proporciona las siguientes ventajas: [25]

- Se pueden desarrollar métodos que resuelvan tareas genéricas sin necesidad del conocimiento del dominio, utilizando los mecanismos de herencia e interfaces de la programación orientada a objetos. Estos métodos podrán ser reutilizados en cualquier sistema CBR (PSM).
- Extender la funcionalidad del armazón y definir nuevas estructuras que contengan datos del dominio consiste en crear clases que implementen dichas interfaces o extiendan las clases abstractas mediante herencia.

Las tareas disponibles al diseñar una aplicación se almacenan en formato XML en el fichero tasks.xml, para añadir nuevas tareas únicamente hay que añadirlas a dicho fichero, aunque se debe tener en cuenta que una tarea no tiene mucha utilidad hasta que no se define un método que la resuelva [25].

A continuación se muestra un ejemplo de definición de la tarea CBR Task: [25]

```
<Task>
    <Name>CBR Task</Name>
    <Description>Main CBR task</Description>
</Task>
```

Fig. 2.11. Ejemplo de Definición de la Tarea CBR Task [25].

Los métodos también se definen en formato XML en el fichero methods.xml, pero su representación es algo más compleja. Definir un método consiste en rellenar los siguientes campos [25]:

- **Nombre:** nombre de la clase que implementa el método (debe implementar la interfaz CBRMethod).
- **Descripción:** descripción del método.
- **Precondiciones:** descripción formal de los requisitos que deben cumplirse para poder aplicar el método.
- **Tipo:** actualmente existen dos tipos de métodos: descomposición y resolución.

- **Parámetros:** define parámetros que necesitan ser configurados para ejecutar el método. Por ejemplo, un método que busque los casos más similares puede solicitar como parámetro la función de similitud que se desea emplear. Se puede emplear como parámetro cualquier objeto que implemente la interfaz CBRTerm que representa la raíz de la ontología CBROnto.
- **Competencias:** tareas que este método puede resolver.
- **Subtareas:** si el método es de tipo descomposición, aquí se enumeran las tareas en las que se descompone la tarea original.
- **Postcondiciones:** descripción formal que define que ha ocurrido al ejecutar este método.

A continuación se muestra un ejemplo de definición del método CBR Method:

```
<Method>
    <Name>jCOLIBRI.method.CBRMethod</Name>
    <Description>Main CBR method that will divide CBR process in the four typical
        tasks.</Description>
    <Type>Decomposition</Type>
    <Competencies>
        <Competence>CBR Task</Competence>
    </Competencies>
    <SubTasks>
        <SubTask>Retrieve Task</SubTask>
        <SubTask>Reuse Task</SubTask>
        <SubTask>Revise Task</SubTask>
        <SubTask>Retain Task</SubTask>
    </SubTasks>
</Method>
```

Fig. 2.12. Ejemplo de Definición del Metodo CBR Method [25].

2.6.3.4 Bases de Casos y Conectores

En jColibri la gestión de la base de casos se divide en dos: el mecanismo de persistencia utilizado para almacenar los casos y la organización en memoria de los casos recuperados [25].

Los casos se pueden almacenar utilizando distintos sistemas de persistencia: ficheros de texto plano, ficheros xml, bases de datos, etc. Para poder abstraerse del mecanismo de persistencia utilizado surge la idea de los conectores. Un conector es cualquier clase que implemente la interfaz Connector y debe saber gestionar los casos almacenados en un determinado sistema de persistencia. Mediante los métodos que define la interfaz Connector podemos recuperar, añadir o eliminar casos sin preocuparnos de si estamos operando sobre ficheros, bases de datos, etc.

Actualmente jColibri ofrece conectores para trabajar con ficheros de texto plano, con bases de datos relacionales y con un motor de lógicas descriptivas denominado RACER [25].

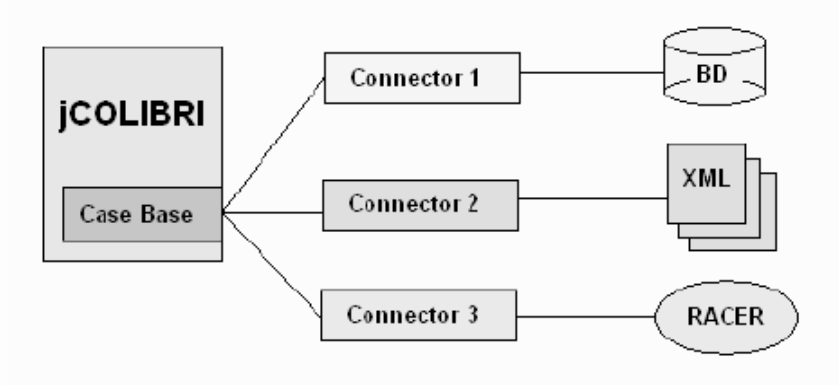


Fig. 2.13. Relación - Base de Casos y Conectores [25].

En segundo lugar, se debe gestionar la organización en memoria de los casos recuperados mediante un conector. Los casos se pueden organizar utilizando distintas estructuras de datos (linear, árboles k-d, case retrieval nets, etc) y esa decisión no debe afectar a la forma en que los métodos acceden a los casos. Para conseguirlo se utiliza la interfaz CBRCaseBase que define el acceso a los casos de manera uniforme. De esta forma podremos indexar los casos utilizando distintos criterios con tan solo escribir distintas clases que implementen dicha interfaz [25].

Esta manera de gestionar los casos utilizando dos niveles de abstracción proporciona mucha flexibilidad. Se puede cambiar el mecanismo de persistencia y la organización en memoria de los casos sin tener que modificar el resto del sistema CBR. Por otra parte, extender la gestión de los casos consiste simplemente en escribir clases que implementen los interfaces correspondientes [25].

2.6.3.5 Representación de los Casos

En jColibri los casos se representan de manera muy general. Un caso se modela como un individuo que se relaciona con otros individuos (los atributos del caso). Esta representación permite construir estructuras arbitrariamente complejas y tratarlas de forma uniforme [25].

Cualquier clase que implemente el interfaz CBRCase será considerada un caso. Actualmente existen dos implementaciones de este interfaz [25]:

- **CBRCaseRecord:** representa casos que se describen como una lista de pares atributo-valor. El caso es un individuo, y sus atributos relaciones hacia otros individuos que contienen los valores de dichos atributos. Esta representación permite definir estructuras jerárquicas en que los atributos de un caso sean otros casos.
- **CBRCaseRacer:** representa casos cuando se utiliza el motor de lógicas descriptivas RACER.

2.6.3.6 Herramienta Gráfica

Aprender a utilizar el Framework, suele implicar un esfuerzo considerable inicialmente. El desarrollador debe aprender muchos conceptos nuevos y conseguir

una visión general de la arquitectura del sistema para poder utilizarlo. Algunas decisiones de diseño pueden resultar difíciles de comprender si no están bien documentadas y quedan implícitas en el código, y las interfaces no suelen ser tan sencillas como se podría esperar [25].

Puesto que jColibri está pensado para ser utilizado por muchas personas distintas, intenta suavizar la curva de aprendizaje inicial del Framework mediante los siguientes medios [25]:

- Documentación que ayuda a entender la arquitectura del sistema rápidamente
- Aplicaciones de ejemplo.
- Una utilidad gráfica que permite crear nuevas aplicaciones visualmente.

La herramienta visual ayuda de dos formas distintas [25]:

- Permite gestionar las tareas y métodos disponibles.
- Permite crear una nueva aplicación seleccionando las tareas que la componen y los métodos que van a resolverlas. Una vez construida la aplicación podemos generar el código java correspondiente y terminar de implementar aquellas características que aun no se pueden diseñar visualmente.

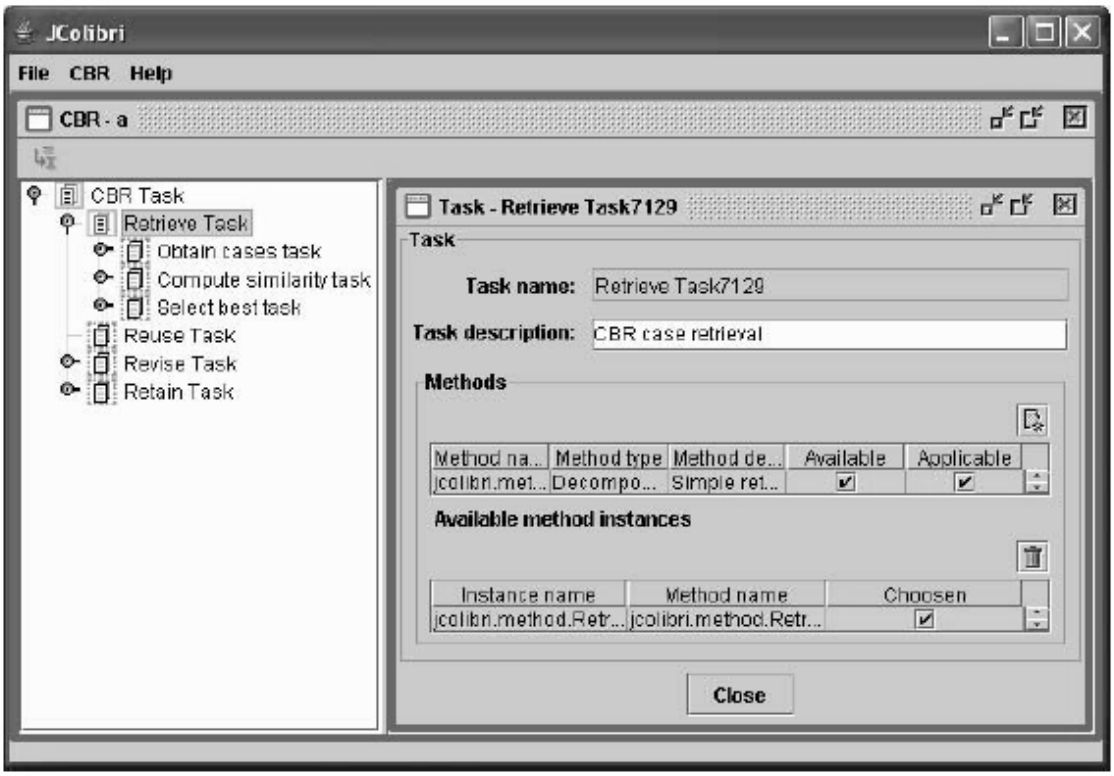


Fig. 2.14. Interfaz de jColibri [25].

2.6.4 Versiones

Existen dos versiones de la plataforma jColibri denominadas jColibri 1 y jColibri 2.

La primera versión del armazón se fundamenta en una arquitectura basada en Métodos de Resolución de Problemas (PSMs) que podían componerse a través de una completa interfaz gráfica y que son capaces de generar el código final de la aplicación RBC. Las herramientas ofrecidas permiten además configurar el acceso a las bases de casos, el diseño de la estructura de los casos y la configuración de las distintas medidas de similitud, así como otras características de los sistemas CBR. Aunque estas herramientas eran muy útiles para usuarios diseñadores que no deseaban entrar en los detalles del código del armazón, no resultaban apropiadas para los desarrolladores que quisieran ampliar la funcionalidad de jColibri extendiendo sus clases y métodos.

Esta y otras razones son las que conducen al desarrollo de jColibri 2. Jcolibri 2 es una plataforma dividida en dos capas: una orientada a desarrolladores y otra orientada a usuarios diseñadores la descripción de jColibri 2 se centra en el diseño de un armazón de caja blanca que facilite su extensión por los programadores y que sirva como base sólida a las herramientas de composición que se ofrecerán a los diseñadores.

2.6.5 jColibri 1

jColibri 1 es la primera versión de esta plataforma para la composición de sistemas CBR. Esta versión permite el desarrollo de aplicaciones CBR mediante una completa interfaz gráfica que posibilita la composición de los distintos métodos incluidos en el armazón. Una vez que el sistema ha sido configurado, es posible generar el código de la aplicación CBR y ejecutarla o integrarla de forma independiente [26].

La arquitectura de jColibri 1 está muy ligada a las ideas de diseño de su predecesora COLIBRI1, creada por Díaz-Agudo [21]. COLIBRI es un armazón para el desarrollo de aplicaciones CBR (más específicamente KI-CBR) implementado mediante el lenguaje LOOM. Sin embargo, el lenguaje de implementación utilizado dificultaba su extensión y popularización. Por ello se realiza su actualización al lenguaje orientado a objetos Java y pasa a denominarse jColibri [26].

La Versión a utilizar en el presente trabajo de investigación es el jColibri1, por lo cual solo se explicara la estructura de esta versión.

2.6.6 Diseño y Arquitectura de jColibri 1

jColibri1 es la evolución tecnológica de la plataforma COLIBRI a los lenguajes de orientación a objetos y nuevas tecnologías de componentes. Actualmente, cualquier herramienta o plataforma de desarrollo pública que quiera conseguir cierta aceptación debe ser implementada en este tipo de lenguajes. En el caso de los armazones este requisito se acentúa debido a su capacidad de extensión: encapsulación, herencia, polimorfismo, etc. [26].

Por todo ello, jColibri se presenta como una plataforma de composición de sistemas CBR donde se aplican y aprovechan todas estas nuevas tecnologías. Aunque el cambio tecnológico es importante, el diseño de esta versión inicial del armazón estuvo muy ligado a las ideas arquitectónicas de su precursor [26].

En la Figura 2.15., se muestran los elementos más importantes de la arquitectura de jColibri 1.

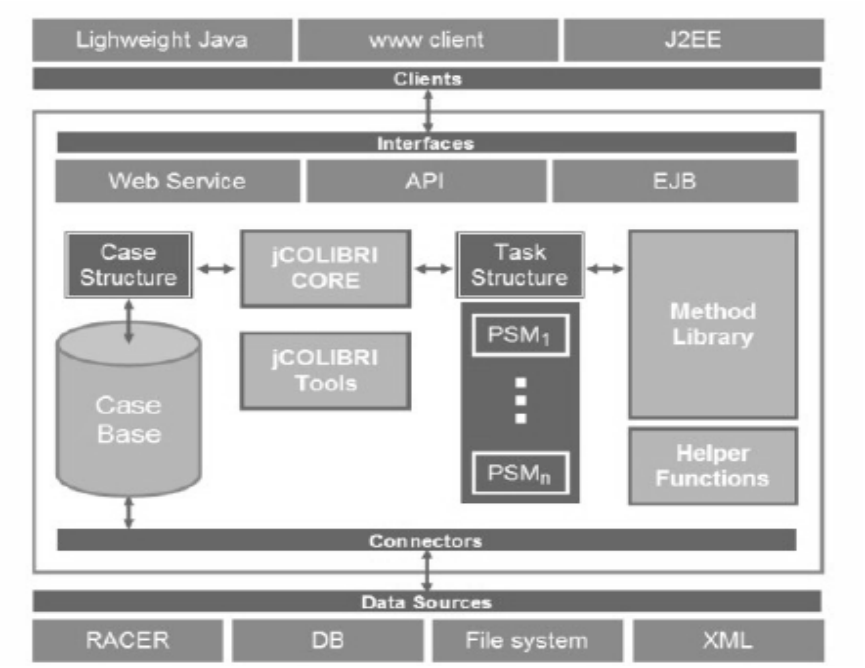


Fig. 2.15. Arquitectura Global de jColibri 1 [30].

La arquitectura de jColibri1 comprende una jerarquía de clases de Java más archivos XML que configurarán las aplicaciones de CBR generadas [30].

2.6.7 Funcionalidad y Características de jColibri 1

Basándose en la arquitectura descrita en la sección anterior, jColibri 1 incorpora los métodos más comunes para el desarrollo de aplicaciones CBR. Además se incluye funcionalidad extra agrupada en forma de extensiones. En esta primera versión de la plataforma los métodos se centran principalmente en el proceso de recuperación de casos, ya que las etapas de adaptación son muy dependientes del dominio concreto de la aplicación. Además, se ofrece una serie de métodos que permiten la evaluación de las aplicaciones generadas con el armazón [26].

Aunque no se ofrezcan detalles sobre los métodos involucrados, a continuación se describirá el proceso de composición de aplicaciones CBR utilizado en esta versión del armazón y el mecanismo de generación de código a partir de las descomposiciones en tareas y métodos [26].

2.6.8 División del Ciclo CBR de jColibri 1

En las etapas iniciales del desarrollo del armazón se encontró un gran problema de eficiencia en las aplicaciones CBR generadas. Esta deficiencia se debía a que el ciclo CBR incluía la carga de los casos en memoria y por lo tanto esta tarea se repetía cada vez que se recibía una consulta. Dicha carga de los casos es una de las tareas computacionalmente más complejas del ciclo CBR, requiriendo normalmente bastante tiempo y recursos. Así que, para resolver este problema se decidió dividir las aplicaciones CBR de jColibri en tres etapas: [26]

- **El preciclo:** Se encarga de cargar los casos e inicializar los recursos necesarios. Esta parte se ejecuta una única vez antes del ciclo principal.

- **El ciclo CBR:** Incluye la funcionalidad normal de un sistema CBR. Puede ser ejecutado varias veces con distintas consultas.

- **El postciclo:** Se encarga de liberar los recursos de la aplicación.

La importancia del preciclo se ha ido corroborando con las sucesivas extensiones del armazón. Como ejemplo más claro podemos citar los métodos de CBR Textual. Este tipo de métodos necesitan procesar el texto para extraer la información y representarla de forma estructurada. Es fácil imaginar que este tipo de proceso puede requerir un largo periodo de cómputo (a veces de varios minutos) por lo que sería inviable ejecutarlo cada vez que se realiza una consulta. Mediante el preciclo se permite adelantar la inicialización y carga de recursos de forma que se ejecuten los procesos más pesados antes de recibir la consulta [26].

El ciclo se encarga de implementar la funcionalidad de cada aplicación CBR. Dentro de esta etapa se ejecuta la recuperación, adaptación y almacenamiento de los casos (aunque algunas de estas tareas puedan ser obviadas) [26].

Por último, el postciclo libera los recursos de la aplicación y ejecuta los métodos de mantenimiento. Estos últimos métodos suelen ejecutarse de forma independiente al ciclo CBR, quizás tras la ejecución de cierto número de ciclos, y se encargan de evitar la degradación en eficiencia del sistema según se incorporen nuevos casos que pudieran añadir ruido. Gracias al postciclo, los métodos de mantenimiento quedan incorporados al ciclo CBR de forma natural sin separarlos del proceso de desarrollo de todo el sistema. Cada una de las tres etapas anteriores se representa como una tarea normal que es resulta por métodos de descomposición en las subtareas apropiadas. Aunque al ejecutar la aplicación estas tres tareas principales se ejecuten independientemente, esta forma de representarlas mantiene la coherencia con el mecanismo de diseño basado en la descomposición de tareas [26].

2.6.9 El Núcleo de jColibri 1

El núcleo (llamado CBRCore en el código fuente) es el componente más importante del Framework. Está a cargo del mantenimiento del CBR, la configuración y ejecutar la aplicación [30].

Cuando un usuario genera una plantilla de aplicación de CBR, se están generando el código de Java que configura realmente un componente del núcleo con las tareas apropiadas, los métodos, tipos de datos, las estructuras de caso, etc. Luego para operar la aplicación, el usuario solamente tiene que llamar al Núcleo de los métodos [30].

El punto principal es compuesto por esto componentes: [30]

- **CBRState:** Mantiene la configuración de las tareas y de los métodos

- **CBRContext: A**ctúa como una pizarra donde los métodos pueden intercambiar los datos. Generalmente, contiene la base de datos y las cajas trabajadoras (dependiendo de la ejecución de los pasos de las cajas de trabajo se pueden recuperar los casos, adaptar los casos, etc.).

- **Packages:** Dirige los componentes remaining: tipos de datos, funciones de Similaridad, estructuras de caso, etc.

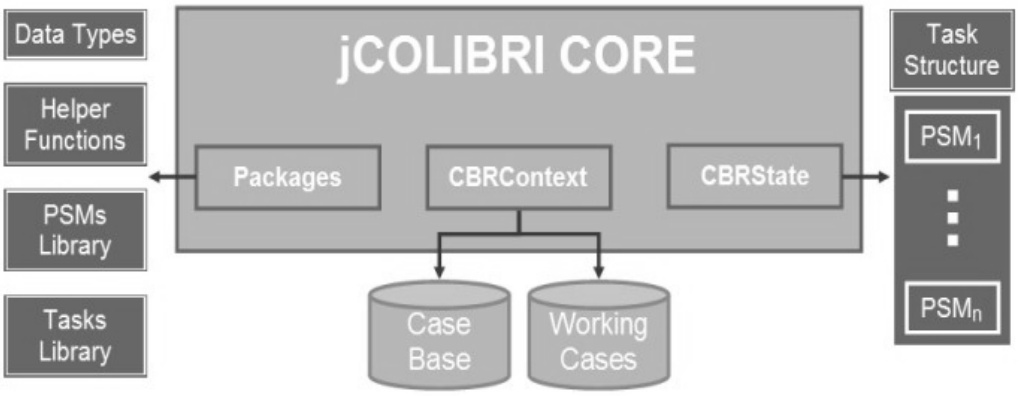


Fig. 2.16. Núcleo de jColibri1 [30].

2.6.10 Proceso de Recuperación de Casos de jColibri1

Esta versión del Framework esta muy ligada al método de recuperación k-NN (k vecinos mas próximos) y por lo tanto permite configurarlo gráficamente. El cálculo del vecino más cercano suele realizarse mediante funciones de similitud locales que obtienen la similitud de cada atributo, y funciones de similitud globales que suelen realizar una media ponderada con los valores devueltos por las medidas locales. Para poder configurar este proceso gráficamente, la interfaz encargada de gestionar la estructura del caso permite asignar funciones de similitud a cada uno de los atributos. Además se permite asignar un peso a cada atributo y fijar los parámetros de las medidas de similitud tal y como se muestra en la Figura 2.17 [26].

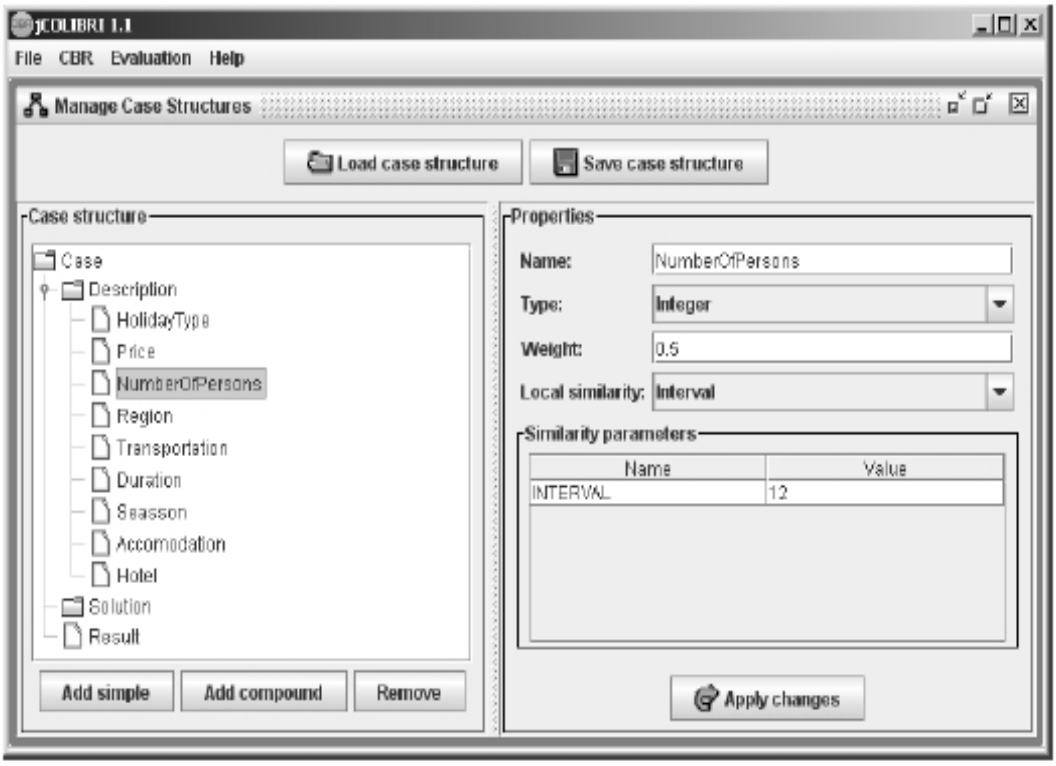


Fig. 2.17. GUI de Configuración de Estructura del Caso [26].

La recuperación es una de las partes más importantes dentro de un sistema CBR y ha sido el centro de gran parte de la investigación. En general, cuanto más eficiente sea la recuperación de casos mejor será la solución obtenida. El algoritmo de recuperación más utilizado es el de los k vecinos más próximos (conocido como k-NN del inglés k Nearest Neighbors). Este algoritmo compara la consulta con las descripciones de los casos y asigna un valor de similitud a cada uno. Seguidamente se recuperan los k casos con similitud más alta [26].

Existen diversas formas de calcular la similitud dependiendo de la representación de los casos. Normalmente los casos se representan como un conjunto de pares atributo-valor por lo que se puede obtener su similitud por medio de dos tipos de funciones. La función de similitud local calcula el parecido de cada uno de los atributos, mientras que la función de similitud global realiza una media ponderada de los valores devueltos por las medidas locales [26].

La clase de los algoritmos de la selección usados por la mayoría de los productos de CBR se llama algoritmo del vecino más cercano [69].

$$Similarity(T,S) = \sum_{i=1}^{n} f(T_i, S_i) \times w_i$$

Fig. 2.18. Algoritmo del Vecino más Cercano [69].

```
COMIENZO
 Entrada: D = {(x_1, c_1), ..., (x_N, c_N)}
     x = (x_1, ..., x_n) nuevo caso a clasificar
 PARA todo objeto ya clasificado (x_i, c_i)
     calcular d_i = d(x_i, x)
 Ordenar d_i(i = 1, ..., N) en orden ascendente
 Quedarnos con los K casos D_x^K ya clasificados
 más cercanos a x
 Asignar a x la clase más frecuente en D_x^K
FIN
```

Fig. 2.19. Representación del Algoritmo K – NN Básico [70].

2.6.11 CBR Textual en jColibri

La extensión de CBR Textual incluida en el jColibri contiene una serie de métodos que permiten el procesamiento de los textos para estructurar su información, ciertas medidas de similitud específicas, y otros componentes auxiliares. Debido a la variedad de las aplicaciones de textual CBR y a la complejidad al manejar la información contenida en textos, se distinguen dos grupos de técnicas para procesar la información: CBR textual semántico y CBR textual estadístico, se describirá el CBR textual semántico para fines de esta aplicación.

2.6.11.1 CBR Textual Semántico.

Los métodos de TCBR semántico extraen información del texto y la representan en los atributos de los casos. Una vez que los casos han sido "rellenados" con esta información, es posible aplicar las técnicas clásicas de CBR que trabajan con casos estructurados. Este tipo de técnicas se califican como semánticas porque intenta "entender" el significado de los textos y organizar su contenido de forma estructurada para poder razonar con él. La gran mayoría de técnicas y sistemas de CBR Textual existentes pertenecen a este grupo [25].

Para el diseño de la extensión textual en jColibri, se eligió el modelo en capas de Lenz [75]. Este modelo se toma como arquitectura de referencia ya que es el diseño más genérico y su flexibilidad permite diferentes combinaciones y extensiones de los diferentes mecanismos de procesamiento de textos.

2.6.11.2 El modelo teórico de Lenz para TCBR

Este modelo se basa en diferentes capas que van procesando los textos gradualmente:

- **Palabras clave (Keyword layer).** Esta capa divide los textos en términos, descarta las palabras vacías (denominadas comúnmente como stopwords: preposiciones, conjunciones,...), y calcula estadísticas sobre la frecuencia de los términos. Esta capa también incluye un análisis gramatical que será utilizado por las capas posteriores. Este paso es

independiente del dominio, por lo que puede ser compartido entre distintas aplicaciones CBR.

- **Expresiones (Phrase layer).** Identifica expresiones específicas del dominio mediante un diccionario. La principal dificultad de esta etapa es que los términos que forman la expresión pueden aparecer de forma no consecutiva en el texto y que las reglas que obtienen las expresiones deben ser generadas manualmente.

- **Tesauro.** Esta capa identifica sinónimos y términos relacionados. Los métodos implementados en esta etapa se utilizan para relacionar los términos de la consulta con los casos. Esta fase es independiente del dominio y para textos en inglés se suele utilizar WordNet, según Miller [68], como tesauro.

- **Glosario.** Esta capa es análoga al tesauro aunque dependiente del dominio concreto de la aplicación. Por lo tanto, es conveniente definir una interfaz común para ambas capas. El principal inconveniente de este paso consiste en la adquisición del glosario.

- **Características (Feature Value).** Esta capa se utiliza en aplicaciones que utilizan casos semiestructurados. Se encarga de extraer características del caso y representarlas en forma de pares <atributo,valor>. Este paso es también específico del dominio. Clasificación en el dominio (Domain Structure). Utiliza las capas anteriores para clasificar los documentos a alto nivel. Esta etapa asigna etiquetas a los casos que pueden ser útiles a la hora de indexarlos y recuperarlos.

- **Extracción de información**. Esta capa se encarga de extraer partes de los textos (o información sobre los mismos) y representarla de forma estructurada. Esta etapa puede solapar con las dos anteriores.

En la mayoría de estas capas se necesita aplicar mecanismos de Procesamiento del Lenguaje Natural que realicen sucesivas transformaciones en los textos hasta prepararlos para extraer y organizar la información. Este tipo de algoritmos suelen ser muy comunes en los campos de Recuperación de Información y Procesamiento de Lenguaje Natural. Por lo tanto existen algunas librerías que contienen implementaciones de los mismos y que pueden ser reutilizadas para desarrollar los métodos del modelo de Lenz [75].

2.7 Protégé

Protégé, es una herramienta de desarrollo de ontologías creada por el grupo Stanford Medical Informatics. Esta herramienta funciona internamente con una representación basada en marcos e implementada en CLIPS. Lleva muchos años en funcionamiento y ha ido formando una comunidad de usuarios muy importante [17].

Se distribuye como una única aplicación Java de código libre que se ejecuta localmente en la máquina del cliente. Sin embargo se comporta como una herramienta híbrida al distribuirse con dos variantes, una que permite representar, cargar y guardar el conocimiento en RDF y otra más moderna llamada Protégé-OWL, que incluye un plug-in para poder hacer lo mismo en OWL. Permite editar y visualizar clases, propiedades, individuos y demás expresiones lógicas posibles mediante un interfaz gráfico muy agradable e intuitivo. Su arquitectura está basada en plug-ins, pensados para realizar funciones muy diferentes. Muchos están pensados para el marcado semántico de documentos, y existen otros que permiten editar y ejecutar reglas SWRL. Permite conectarse fácilmente con razonadores de lógicas descriptivas que acepten la interfaz de comunicación DIG. Protégé proporciona una potente API llamada Protégé-OWL API, para trabajar con las tecnologías asociadas a los lenguajes RDF y OWL, cargar, salvar, consultar y realizar modificaciones en bases de conocimiento que sigan estos modelos, así como comunicarse con razonadores de lógicas descriptivas. Este API puede utilizarse para desarrollar plug-ins que se ejecuten como componentes integrados en la interfaz de usuario de Protégé o para desarrollar aplicaciones independientes [17].

En cuanto a visualización de ontologías y bases de conocimiento, Protégé cuenta con un plug-in llamado Jambalaya que permite representar y editar gráficamente los ficheros que está creando el usuario. También ofrece la posibilidad de generar automáticamente documentación para ontologías y bases de conocimiento en formato OWLDoc [17].

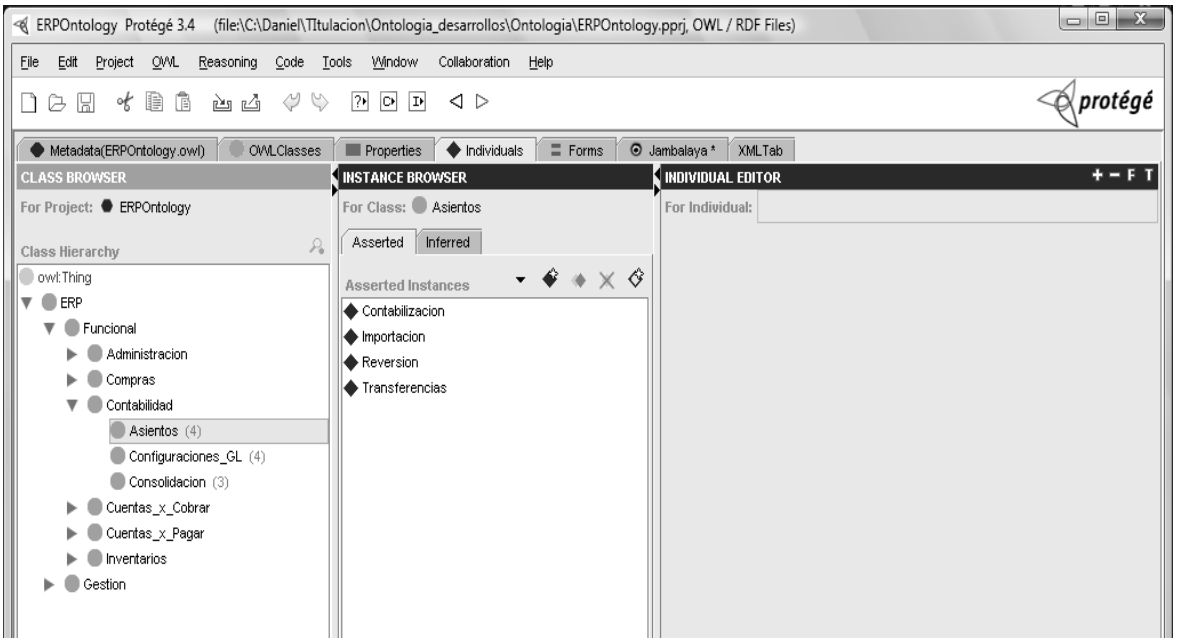


Fig. 2.20. GUI del Editor de Ontologías Protégé.

2.8 Modelo para la Representación de una Memoria Organizacional en Base a Casos

El conocimiento que posee una organización es muy grande, por lo que deberá seleccionarse primeramente aquel que sea verdaderamente importante y relevante para la competitividad de la empresa. La organización requiere de conocerse a sí misma, identificar qué es lo que mejor hace, cómo lo hace y cómo integrar esas capacidades para crear sus ventajas competitivas y diferenciación ante la competencia [62].

Es de vital importancia tener en la memoria organizacional aquel conocimiento que permita contestar las siguientes preguntas: ¿Por qué se hizo esto de cierta manera?, ¿Este problema no ha sido resuelto antes?, ¿Alguien ha tratado de considerar este nuevo enfoque? y ¿Qué aprendimos la última vez que sucedió ese problema? [64].

Como se ha visto, almacenar en la memoria organizacional experiencias y conocimientos pasados es muy importante. Una de las maneras de lograrlo, es guardar esta experiencia por medio de casos. La razón de enfocarse a casos, se debe a que a través de ellos puede transferirse gran parte del conocimiento que actualmente no se tiene documentado y que contiene experiencias, habilidades y competencias del trabajo realizado por el personal de la empresa que le da a la organización ventajas competitivas [64].

2.8.1 Casos

Un caso es una pieza contextualizada de conocimiento que representa una experiencia. Contiene la lección pasada que es el contenido del caso y el contexto en el cual la lección puede ser utilizada [60]. Típicamente un caso comprende:

- **El problema** que describe el estado del mundo cuando ocurrió el caso.
- **La solución** que establece la solución derivada de ese problema, y/o
- **El resultado** que describe el estado del mundo después de ocurrido el caso.

De acuerdo a lo anterior, un caso se puede definir como la descripción detallada de una experiencia del pasado sobre una situación particular, formada por la descripción del problema, la solución tomada para resolver el problema y el resultado obtenido después de aplicada la solución (Ver Figura 2.21.) [64].

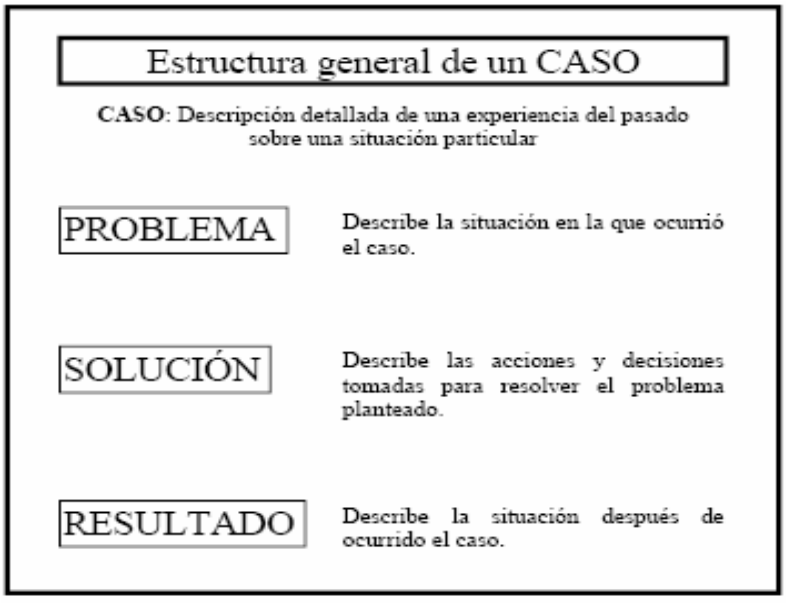


Fig. 2.21. Estructura General de un Caso [57].

Los casos que incluyen un problema y solución, por ejemplo, pueden usarse para derivar soluciones a nuevos problemas. Si además tienen una descripción de la situación y resultado, pueden usarse para evaluar nuevas situaciones. El caso que tiene una solución

especificada puede usarse en la evaluación de propuestas de solución y anticipar problemas potenciales antes de que ellos ocurran [60].

El tipo de casos que desean documentarse para el propósito de este trabajo, son aquellos cuyo contenido en el problema, solución y resultado pueda expresarse como una secuencia de eventos [64].

2.8.1.1 Contenido del Problema

El problema o descripción de la situación del caso, contiene el estado del mundo cuando surgió la problemática. Está representado por un problema que necesita resolverse o una situación que necesita ser interpretada, clasificada o comprendida. En general, el tomador de decisiones determina cuál caso es aplicable a una nueva situación examinando las similitudes entre las descripciones del nuevo problema con la situación anterior. Si una situación es suficientemente similar a la descripción de un problema pasado, el caso es seleccionado. Por lo tanto, el problema debe tener suficiente detalle para que pueda evaluarse la aplicabilidad del caso a una nueva situación.

2.8.1.2 Contenido de la Solución

La solución es, en pocas palabras, los conceptos u objetos que logran alcanzar el conjunto de objetivos que se plantearon en la descripción del problema, tomando en cuenta las restricciones especificadas y otras características contextuales del problema [60].

2.8.1.3 Contenido del Resultado

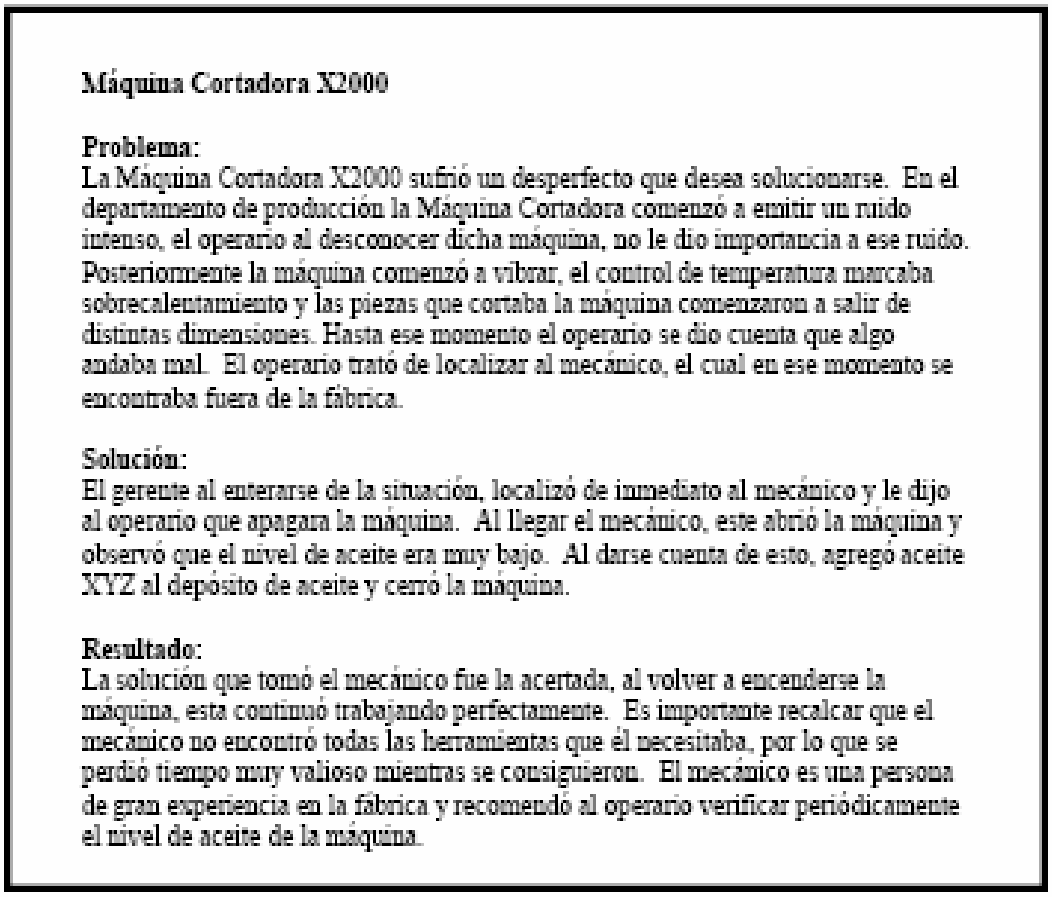
El resultado de un caso especifica lo que sucedió como resultado de llevar a cabo la solución. El resultado incluye la retroalimentación y la interpretación de esa retroalimentación. Con esta información, el tomador de decisiones puede anticipar problemas potenciales y predecir resultados al proponer una solución.

2.8.2 Beneficios

La finalidad de conocer una gran variedad de casos que describen situaciones pasadas, es la de darle una ventaja al tomador de decisiones al momento de enfrentarse a un nuevo problema [58]. Entre más detallada sea la información contenida en el caso, mejor utilidad tendrá en el apoyo a la solución de nuevas situaciones [64].

2.8.3 Ejemplo de un Caso

Considerando todos los criterios mencionados en los puntos anteriores, en la Figura 2.22, se muestra un ejemplo de la estructura y contenido básico de un caso [64].



Máquina Cortadora X2000

Problema:
La Máquina Cortadora X2000 sufrió un desperfecto que desea solucionarse. En el departamento de producción la Máquina Cortadora comenzó a emitir un ruido intenso, el operario al desconocer dicha máquina, no le dio importancia a ese ruido. Posteriormente la máquina comenzó a vibrar, el control de temperatura marcaba sobrecalentamiento y las piezas que cortaba la máquina comenzaron a salir de distintas dimensiones. Hasta ese momento el operario se dio cuenta que algo andaba mal. El operario trató de localizar al mecánico, el cual en ese momento se encontraba fuera de la fábrica.

Solución:
El gerente al enterarse de la situación, localizó de inmediato al mecánico y le dijo al operario que apagara la máquina. Al llegar el mecánico, este abrió la máquina y observó que el nivel de aceite era muy bajo. Al darse cuenta de esto, agregó aceite XYZ al depósito de aceite y cerró la máquina.

Resultado:
La solución que tomó el mecánico fue la acertada, al volver a encenderse la máquina, esta continuó trabajando perfectamente. Es importante recalcar que el mecánico no encontró todas las herramientas que él necesitaba, por lo que se perdió tiempo muy valioso mientras se consiguieron. El mecánico es una persona de gran experiencia en la fábrica y recomendó al operario verificar periódicamente el nivel de aceite de la máquina.

Fig. 2.22. Ejemplo de un Caso [57].

2.8.4 Modelo

En la Figura 2.23., se muestra el modelo propuesto para la representación del conocimiento inmerso en casos. Este modelo está formado por tres partes principales: Casos, Representación formal de casos y su Representación por medio de herramientas computacionales [64].

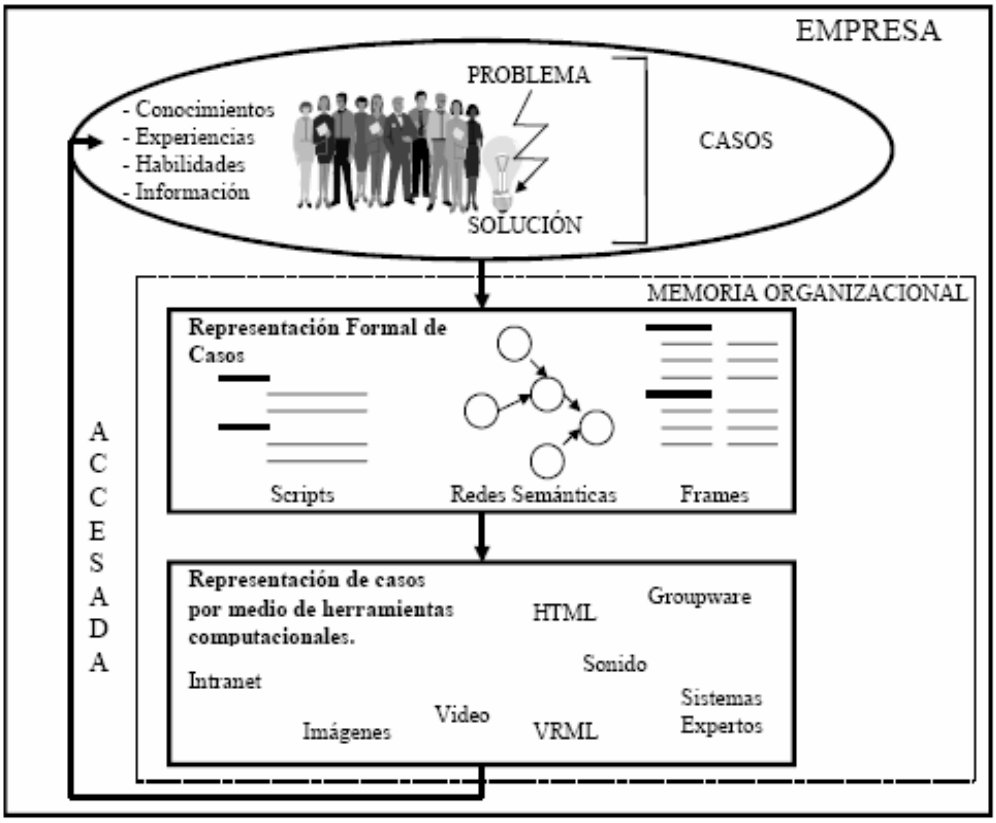


Fig. 2.23. Modelo de Representación de una Memoria Organizacional [57].

La primera parte del modelo, casos, especifica el tipo de conocimiento que interesa representar. Este conocimiento está en forma de caso y tiene todas aquellas experiencias pasadas que han permitido resolver problemas en distintas áreas de la empresa. Estos casos deberán pasar a formar parte de una memoria organizacional compuesta por la representación formal de casos y la representación por medio de herramientas computacionales [64].

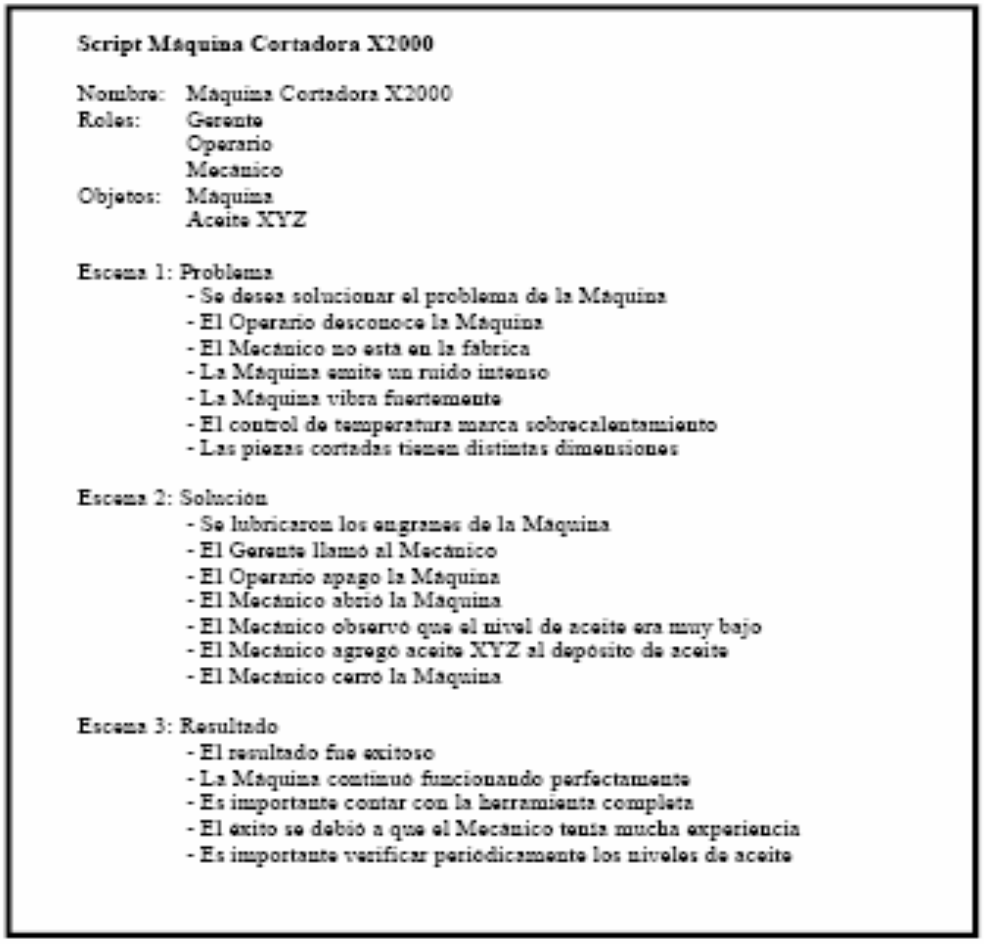
La finalidad de que los casos se documenten de manera formal, es para facilitar su entendimiento y manipulación. Cuando se ha definido la representación formal de los casos, deberá pasar a la tercera parte del modelo en la que se buscará la representación por medio de herramientas computacionales. Esta representación deberá ser la más adecuada que cumpla con dos características, (1) deberá ser una herramienta computacional que permita la captura y facilite el acceso de los casos por todos los empleados sin importar si se encuentran dentro de las mismas instalaciones de la empresa o en alguna sucursal en otra parte del mundo y (2) deberá facilitar la implementación de la representación formal seleccionada [64].

La flecha que va de la representación en herramientas computacionales a la primera parte del modelo, significa que los casos representados en la herramienta computacional sirven para facilitar el acceso del conocimiento inmerso en ellos que fortalezca y desarrolle nuevos conocimientos, experiencias y habilidades en el personal de la empresa para que contribuyan en el mejoramientos de la competitividad de la organización [64].

2.8.5 Representación formal

Después de conocer la definición de un caso, sus características, estructura y contenido, se cuentan con los elementos suficientes para determinar la representación formal que mejor se acople a un caso. Entre las distintas formas de representar formalmente el conocimiento, la que mejor cumple los requisitos es el script. Avron Barr [59] dice que los scripts describen una secuencia de eventos y que permiten entender situaciones para guiar en la interpretación de ocurrencias en situaciones similares [64].

Esto encaja perfectamente con la definición de caso dada por Christopher Riesbeck [58] que lo define como una historia única llena de detalles y que contiene experiencias relevantes sobre una situación particular. Además, el tipo de casos que interesa documentar, contendrán una secuencia de eventos inmersos en el problema, solución y resultado. En la Figura 2.24., se muestra un ejemplo para la representación en script para el ejemplo del caso explicado en la Figura 2.22 [64].



Script Máquina Cortadora X2000

Nombre: Máquina Cortadora X2000
Roles: Gerente
Operario
Mecánico
Objetos: Máquina
Aceite XYZ

Escena 1: Problema
- Se desea solucionar el problema de la Máquina
- El Operario desconoce la Máquina
- El Mecánico no está en la fábrica
- La Máquina emite un ruido intenso
- La Máquina vibra fuertemente
- El control de temperatura marca sobrecalentamiento
- Las piezas cortadas tienen distintas dimensiones

Escena 2: Solución
- Se lubricaron los engranes de la Máquina
- El Gerente llamó al Mecánico
- El Operario apagó la Máquina
- El Mecánico abrió la Máquina
- El Mecánico observó que el nivel de aceite era muy bajo
- El Mecánico agregó aceite XYZ al depósito de aceite
- El Mecánico cerró la Máquina

Escena 3: Resultado
- El resultado fue exitoso
- La Máquina continuó funcionando perfectamente
- Es importante contar con la herramienta completa
- El éxito se debió a que el Mecánico tenía mucha experiencia
- Es importante verificar periódicamente los niveles de aceite

Fig. 2.24. Ejemplo de un Caso Representado en Script [57].

2.8.6 Representación en herramienta computacional

Actualmente existe en el mercado software diseñado para documentar conocimiento inmerso en casos y que cumple perfectamente con el propósito de ser herramientas computacionales que permitan compartir el conocimiento entre todos los miembros de la organización sin importar el lugar geográfico donde se encuentren y que incluyen dentro

de sus algoritmos de búsqueda y recuperación, los conceptos de razonamiento basado en casos y scripts [64].

Entre estas herramientas están el Internet Knowledge Kiosk, CasePoint WebServer, CBR Express [57], etc.

Se combinan tecnologías y herramientas de dominio público que sean factibles y que tengan un costo mínimo. Se puede elegir herramientas como la llamada EWS para utilizar como software de búsqueda de casos, archivos en HTML (Hyper Text Markup Language) para la documentación de casos y páginas Web donde se estructurarán los casos. En la Figura 2.25., se muestra el modelo, las opciones factibles para la representación formal y las tecnologías computacionales con las que se puede implementar este modelo [64].

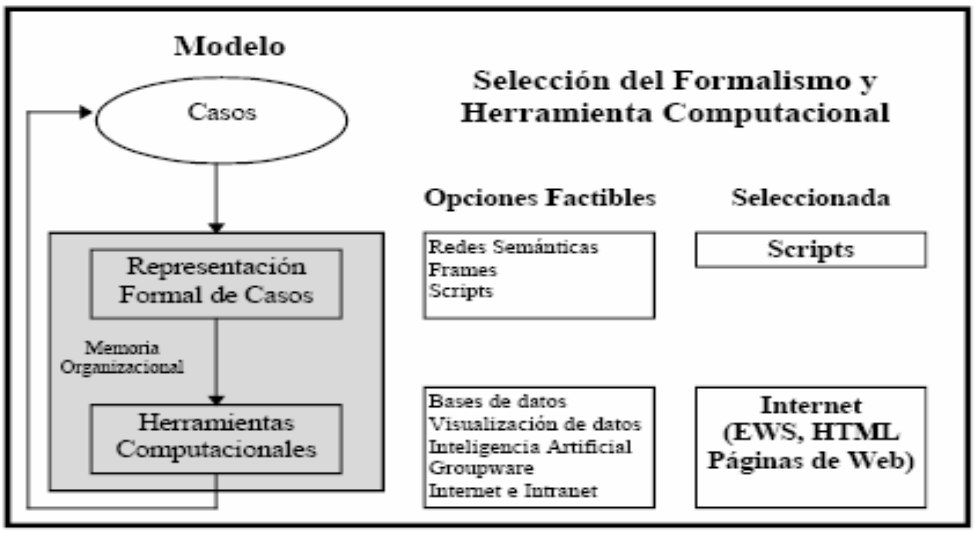


Fig. 2.25. Selección de Formalismo y Herramientas Computacionales [57].

2.8.7 Metodología para la Implementación de una Memoria Organizacional en Base a Casos

Después de planteado el modelo para la representación de la memoria organizacional en base a casos, ahora se describe la metodología para llevar a cabo su implementación en una empresa. Esta metodología consiste en una serie de etapas que contemplan desde la evaluación de la factibilidad de que la empresa desarrolle su memoria organizacional hasta la utilización de la herramienta que implementa la memoria organizacional [64].

2.8.7.1 Etapas de la metodología

Las etapas que deben seguirse para implementar la memoria organizacional basada en casos se muestran en la Figura 2.26., donde también se incluyen las herramientas que pueden apoyar en la ejecución de cada una de las etapas [64].

Etapas de la Metodología	Objetivo	Herramienta
Identificar la factibilidad de la empresa	Determinar la factibilidad de que la organización desarrolle su memoria organizacional.	- Entrevistas.
Identificar el(los) proceso(s) clave	El proceso que contenga conocimientos que sean de utilidad almacenarse en una memoria organizacional.	- Entrevistas. - Reuniones de trabajo.
Identificar conocimiento y los expertos	Identificar conocimientos del proceso y los expertos que pueden aportar sus experiencias.	- Entrevistas. - Reuniones de trabajo. - Cuestionario.
Recopilar y estructurar el conocimiento.	Recopilar los conocimientos para estructurarlos en forma de caso: *problema*, *solución* y *resultado*.	- Reuniones de trabajo. - Estructura del caso. - Cuestionario - Grabadora
Aplicar el modelo propuesto	Representar el caso en forma de script y capturar el caso en la herramienta computacional.	- Formato de un script - El caso - Páginas de Web - EWS.(Excite Web Server)

Fig. 2.26. Metodología para Desarrollar Memorias Organizacionales en Base a Casos [57].

2.8.7.2 Procedimiento de implementación de la Metodología

Para llevar a cabo la implementación de cada etapa de la metodología, se deben realizar las siguientes actividades: [57]

- **Etapa 1:** Identificar la factibilidad de que la empresa desarrolle su memoria organizacional.

  Una empresa que sea factible para desarrollar una memoria organizacional debe contar con las siguientes características [64].

  1. La alta dirección debe tener un interés por el desarrollo de una memoria organizacional.
  2. La organización debe tener conocimientos documentados, en expertos, en personal de mucha experiencia y que desea compartir entre todos sus miembros.
  3. La empresa cuenta con personal con deseos de adquirir y compartir sus conocimientos.

- **Etapa 2:** Identificar el(los) proceso(s) clave de la empresa.

  Se debe hacer una evaluación de todos aquellos procesos de la organización y seleccionar aquellos que mejor cumplan con las siguientes características: [64]

  1. Debe ser un proceso clave para la organización.
  2. Tiene impacto y da un valor agregado a la empresa.
  3. Permite satisfacer los requerimientos del cliente.

4. Balancea de forma efectiva los recursos humanos y tecnológicos de la empresa. Deben asignarse los mejores recursos humanos y tecnológicos a los procesos clave.

- **Etapa 3:** Identificar el conocimiento y los expertos.

  Los pasos a seguir para identificar el conocimiento y los expertos son los siguientes: [64]

  1. Hacer un listado de aquellas situaciones pasadas que han dejado experiencias útiles.
  2. Seleccionar a la(s) persona(s) de mayor experiencia que ha(n) participado directamente en estas situaciones para que aporte su experiencias.
  3. Identificar el conocimiento útil de las situaciones pasadas: descripción detallada de problemas y sus soluciones; habilidades o capacidades que han desarrollado los miembros más antiguos al realizar su trabajo; estrategias o ideas que se han aplicado en situaciones difíciles, cómo le han hecho, qué resultados han tenido; historias de éxito y fracaso.

- **Etapa 4:** Recopilar y estructurar el conocimiento en forma de caso.

  Los pasos a seguir para recopilar y estructurar el conocimiento en forma de caso son los siguientes: [64]

  1. Recopilar el conocimiento del experto.
  2. Estructurar el conocimiento en forma de caso.
  3. Verificar el contenido del caso con el experto.

- **Etapa 5:** Aplicar el modelo propuesto.

  a) Representar el caso en forma de script [64].

  Los pasos a seguir para representar el caso en forma de script son los siguientes:

  1. Representar el caso siguiendo el formato de representación de un script mostrado en la Figura 2.24.

  b) Capturar el caso en la herramienta computacional [64].

CAPITULO III

ESTADO DEL ARTE

3.1 Aplicativos – Frameworks CBR

jColibri es un armazón para construir aplicaciones de razonamiento basado en casos. También se han visto otros dos sistemas que tienen objetivos similares: CBR*Tools y Orange, y se describen a continuación.

3.1.1 CBR*Tools

CBR*Tools es un Framework para crear sistemas CBR desarrollado por Michel Jaczynski y Brigitte Trousse [76]. Sigue un diseño orientado a objetos y esta implementado en Java.

Crear una aplicación CBR, es decir instanciar el Framework, implica especializar los puntos de anclaje predefinidos. Podemos agrupar estos puntos de anclaje en las siguientes categorías: delegación de los procesos de razonamiento, separación del almacenamiento de los casos de su indexación, diseño de índices como componentes reutilizables, y patrones de diseño de adaptación.

Con una filosofía similar a jColibri, CBR*Tools utiliza interfaces que sirven como puntos de anclaje del Framework. Podemos especializar el comportamiento de la aplicación implementando estos interfaces.

3.1.1.1 Delegación de los Procesos de Razonamiento

Como ya hemos comentado, el ciclo principal de un sistema CBR puede descomponerse en cuatro pasos: recuperación, reutilización, revisión y aprendizaje. CBR*Tools propone delegar cada uno de estos pasos en un objeto diferente que implemente el interfaz correspondiente: Recuperar, Reutilizar, Revisar o Retener.

También define un interfaz Reasoner que deberá implementar el objeto encargado de gestionar el proceso de razonamiento (ejecución de la aplicación): comenzándolo, parándolo o continuándolo. Los diferentes pasos que componen la aplicación se

comunican mediante un contexto de ejecución que se hace explícito mediante el objeto Reasoning y almacena el estado de razonamiento.

Todos estos objetos se obtienen a través de un objeto factoría llamado ReasonerFactory, lo que proporciona las siguientes ventajas:

- Independencia entre el control de razonamiento y su implementación.
- Reutilización de implementaciones alternativas a una misma interfaz.
- Aísla el código necesario para combinar los procesos que intervienen en el ciclo CBR.
- Garantiza la combinación correcta de los procesos, desde el punto de vista de los interfaces.

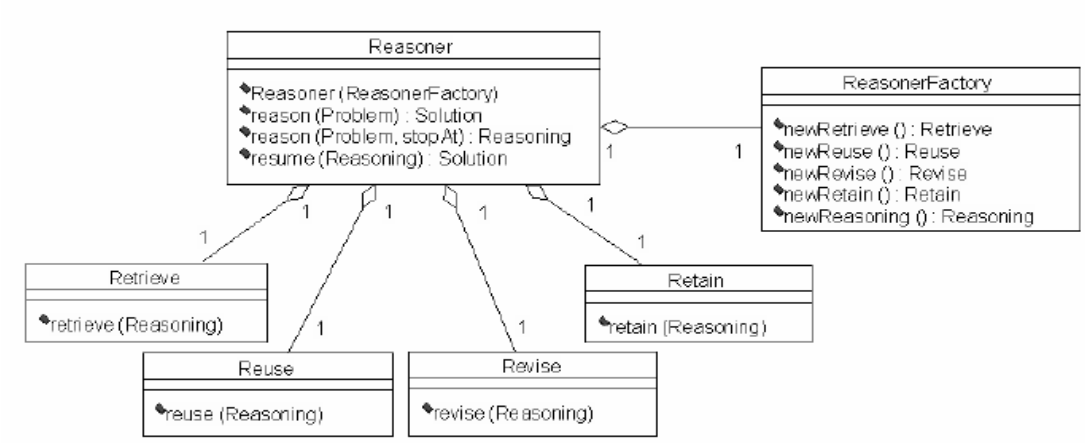


Fig. 3.1. Modelo de Objetos de CBR*Tools [25].

3.1.1.2 Separación del Almacenamiento de los casos de su indexación

En CBR*Tools se separa el almacenamiento de los casos de los mecanismos de indexación utilizados. Podríamos tener incluso varias estructuras de indexación sobre los mismos casos para acelerar distintos tipos de búsquedas.

Los objetos Reasoner acceden a los casos a través de la clase Memory que abstrae de los mecanismos internos de acceso a los casos. La clase Memory utiliza otros dos objetos para gestionar la base de casos: CaseBase e IndexBase. CaseBase se encarga de acceder a los casos, incluyendo los mecanismos necesarios para acceder a bases de casos distribuidas, etc. IndexBase se encarga de organizar los casos según distintos criterios para acelerar el acceso (indexación).

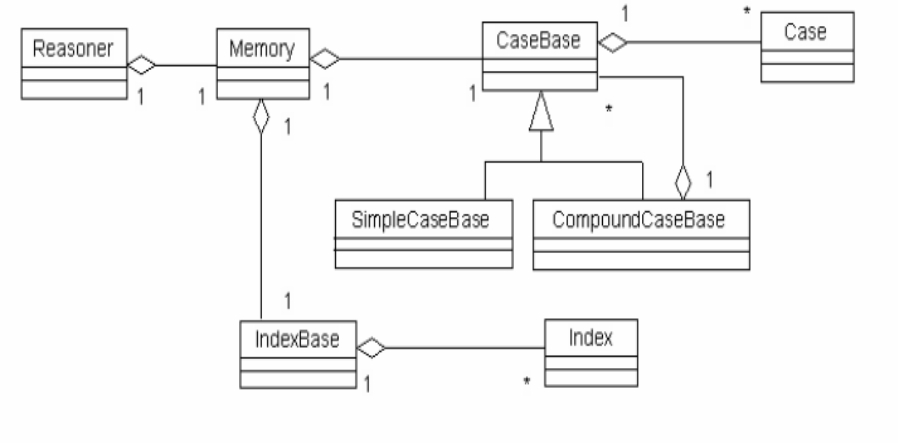


Fig. 3.2. Estructura de Indexación de CBR*Tools [25].

3.1.1.3 Índices como componentes reutilizables

Los índices utilizados para acelerar el acceso a la base de casos según distintos criterios siguen el patrón Composite, por lo que se pueden componer para construir estructuras más complejas. La idea de CBR*Tools es poder utilizar la indexación adecuada en cada subespacio de casos de la base de casos.

3.1.1.4 Patrones de Adaptación

CBR*Tools también considera la idea de proporcionar una biblioteca de patrones de adaptación que puedan ser seleccionados y especializados por las aplicaciones.

3.1.2 Orenge

Empolis Orenge es una plataforma comercial para el desarrollo de sistemas CBR.

Sus principales ventajas son las siguientes:

- Plataforma basada en componentes. Los componentes se organizan en pipelines que describen el flujo de control. Se pueden añadir nuevos componentes específicos de la aplicación fácilmente.
- Incluye potentes componentes por defecto para recuperar e indexar los casos. Proporciona también un componente para minería de texto para extracción de información.
- Se pueden recuperar casos desde distintas bases de datos y múltiples formatos de fichero. Las consultas se realizan en formato XML y las respuestas se obtienen en el mismo formato.
- La plataforma es muy escalable, lo que permite crear aplicaciones que gestionen miles de documentos.

El componente principal Orenge es el ProcessManager y gestiona el resto de servicios. Cada componente utiliza conocimiento distribuido en un conjunto de archivos siguiendo el modelo de contenedor de conocimiento presentado por Ritcher [77].

Ritcher [77], identificó los siguientes contenedores para los sistemas CBR: el vocabulario, las medidas de similitud, el conocimiento de adaptación, y la base de casos.

La comunicación entre los diferentes procesos configurados en el flujo de ejecución se hace a través de un mecanismo de pizarra, donde cada proceso que se ejecuta puede leer los datos disponibles en esta y debe escribir el resultado de su ejecución, para que esté disponible a los procesos siguientes.

3.1.2.1 Recuperación:

La base de casos se puede gestionar de tres formas diferentes:

- Recupera todos los casos y los mantiene en memoria.
- Los casos se almacenan en una base de datos y se recuperan mediante consultas SQL.
- En memoria sólo se mantiene la estructura de indexación de los casos (basada en Case Retrieval Nets) y cuando es necesario recuperar un caso concreto se accede a la base de datos.

Además, como ya hemos comentado, se dispone de un componente de minería de datos que permite construir casos a partir de documentos de texto. Este componente permite trabajar con documentos y consultas en lenguaje natural.

3.1.2.2 Adaptación:

El conocimiento de adaptación se expresa en forma de reglas que determinan cuando es necesario realizar la adaptación y las acciones que se deben realizar. También existe la posibilidad de adaptar la consulta realizada antes de iniciar el proceso CBR para mejorar los resultados.

Por último existen componentes que permiten al sistema interactuar con el usuario y pedirle que valores los resultados, modifique la consulta, generar explicaciones de los resultados obtenidos, etc.

3.1.3 jColibri

jColibri es la evolución de un sistema anterior denominado COLIBRI. Este sistema se centraba en sistemas CBR con conocimiento intensivo (K-I CBR). Estos sistemas se caracterizan por utilizar conocimiento adicional acerca del dominio sobre el que operan lo que, en teoría, debería mejorar su eficiencia.

jColibri es un marco orientado a objetos usado para desarrollar sistemas CBR. Ofrece un fácil proceso de desarrollo que se basa en la reutilización de los últimos diseños e implementaciones. jColibri formaliza el uso del conocimientos de dominio CBR independiente de la ontología (CBROnto), que se asigna en las clases del Framework, un conocimiento a nivel de descripción de las tareas CBR y una biblioteca reutilizable de Métodos de Resolución de Problemas (PSMS) .

La Figura 3.3., muestra la arquitectura de jColibri.

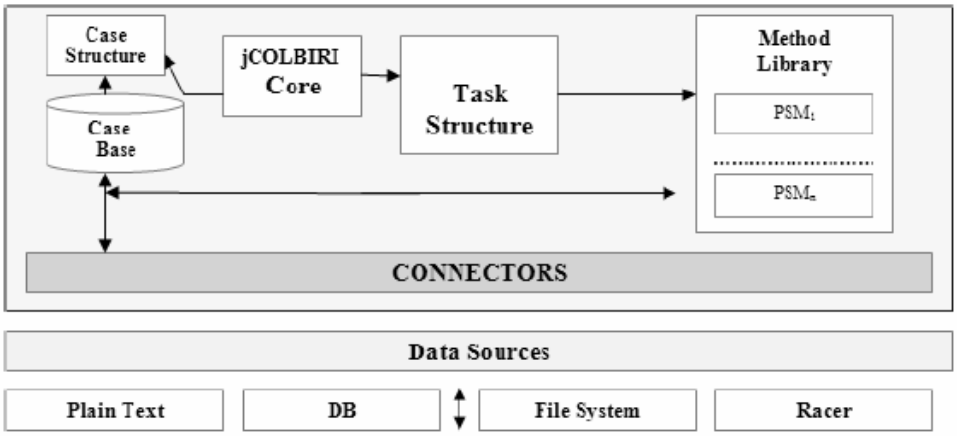


Fig. 3.3. Arquitectura del Framework jColibri [46].

Consta de cuatro módulos principales:

3.1.3.1 Estructura de caso

jColibri representa un caso de una manera muy general. Un caso se modela como un individuo que se relaciona con otros individuos (los atributos del caso). Esta representación permite construir estructuras arbitrariamente complejas y tratarlas de forma uniforme. El Framework soporta varias estructuras de casos, a partir de claros valores de atributos registrados para árboles jerárquicos con atributos compuestos.

Un caso se compone de Descripción (describe el problema por medio de varios atributos), Solución (contiene la descripción de la solución del caso) y Resultados (representa el resultado de aplicar el caso en una situación real). Descripción y solución son conjuntos de atributos, existen dos tipos de atributos: simples y compuestos. Atributos simples son descritos por nombre, tipo, peso y función de la similitud local. Atributos compuestos reúne otros atributos simples, permitiendo la estructura compleja de los casos. Cuando se comparan dos casos, se utilizan las funciones de similitud para comparar valores de los atributos simples. Funciones de similitud global están vinculadas a las características y se utilizan para recopilar similitudes de los atributos compuestos en un único valor de similitud. Finalmente el valor de la similitud de los dos casos se calcula como la similitud de la descripción de sus conceptos.

3.1.3.2 Base de Casos y Conectores

En jColibri la gestión de la base de casos se divide en dos: el mecanismo de persistencia utilizado para almacenar los casos y la organización en memoria de los casos recuperados.

La capa de persistencia está integrada por varios conectores que permiten a los desarrolladores cambiar el tipo de almacenamiento de datos muy fácilmente. Los casos se pueden almacenar utilizando distintos sistemas de persistencia: ficheros de texto plano, ficheros xml, bases de datos, etc. Los Conectores son objetos que saben cómo

acceder y recuperar los casos del medio y retornar estos casos al sistema CBR de manera uniforme. jColibri implementa Texto sin conector, conector JDBC, Conector para sistema de archivos y el Conector Racer.

La segunda capa de la organización en memoria de la base de Casos, es la Base de Casos (estructura de datos) que se utiliza para organizar los casos, una vez leído y cargado por el conector en la memoria.

Esta manera de gestionar los casos utilizando dos niveles de abstracción proporciona mucha flexibilidad. Se puede cambiar el mecanismo de persistencia y la organización en memoria de los casos sin tener que modificar el resto del sistema CBR. Por otra parte, extender la gestión de los casos consiste simplemente en escribir clases que implementen los interfaces correspondientes.

3.1.3.3 Tareas/Métodos Ontología

jColibri ha sido construido en torno a la idea básica de separar tareas (tasks) y métodos (methods). Las tareas indican objetivos que el sistema debe alcanzar y guían la ejecución de la aplicación. Los métodos indican como resolver las tareas.

El más conocido, el análisis a nivel de conocimiento aplicado a los sistemas CBR describe el ciclo general del CBR en términos de cuatro tareas al más alto nivel de generalidad:

- Recuperar la mayoría de caso similar/similares.
- Reutilización de su/sus conocimientos para resolver el problema.
- Revisar la propuesta de solución.
- Retener la experiencia.

Cada una de las tareas del CBR implica una serie de subtareas más específicas. jCOLIBRI incluye nuevos métodos para resolver la preparación y las tareas de mantenimiento, llamadas PreCycle (carga de los casos de la fuente de datos) y PostCycle (almacena los casos aprendidos en la capa de persistencia).

Por lo tanto una aplicación CBR puede representarse como una estructura en forma de árbol con las tareas que deben resolverse. Ejecutar una aplicación consiste en resolver dichas tareas ejecutando los métodos correspondientes.

3.1.3.4 Núcleo jColibri

El núcleo es el componente más importante del Framework. Este se encarga del mantenimiento de la configuración del CBR y la ejecución de la aplicación. Cuando un usuario genera una plantilla de aplicación CBR, se esta generando el código Java que configura los componentes del núcleo con las adecuadas tareas, métodos, tipos de datos y estructuras caso. El núcleo se compone de CBRState (mantiene las tareas y métodos de configuración), CBRContext (contiene la Base de Casos y casos

trabajados), Paquete (gestiona los componentes restantes, tales como las funciones de similitud y las estructuras de casos).

La herramienta visual del Framework ayuda de dos formas distintas:

- Permite gestionar las tareas y métodos disponibles.
- Permite crear una nueva aplicación seleccionando las tareas que la componen y los métodos que van a resolverlas. Una vez construida la aplicación podemos generar el código java correspondiente y terminar de implementar aquellas características que aun no se pueden diseñar visualmente.

3.1.4 Evaluación Comparativa

A continuación haremos evaluación de los Frameworks CBR*Tools y Orenge, comparando ambos con JColibri.

- CBR*Tools es un marco orientado a objetos, implementado en Java, diseñado para facilitar el desarrollo de aplicaciones CBR. Se identifican los siguientes ejes de variabilidad: las medidas de delegación de razonamiento, la separación del espacio de almacenamiento de los casos y la indexación de los casos, el diseño de índices como componentes reutilizables, y el diseño de patrones de adaptación. El Framework se centra principalmente en la indexación, proporcionando un gran número de alternativas de indexación.
- La diferencia clave de JColibri con respecto a la CBR * Tools es el explícito modelo de tareas o método de descomposición que impone un nivel superior sobre la arquitectura del Framework, facilitando el uso del Framework y su evolución. Sin embargo, en un cierto sentido, en CBR * Tools el nivel de conocimiento es también explícito como parte de su modelo basado en UML. Además, JColibri incorpora una interfaz gráfica para aliviar el esfuerzo de instanciacion del Framework que está basada en su tarea o método de descripción (a nivel del conocimiento).
- La arquitectura de Orenge también está relacionada con JColibri. Orenge ha sido diseñado como una plataforma basada en componentes. Construir una aplicación incluye la selección y composición de los componentes necesarios. La estructura de la secuenciación de sus procesos es muy similar a la biblioteca de métodos de resolución de problemas (PSMS) igual que JColibri.
- De la misma manera que JColibri, Orenge también proporciona acceso a bases de datos y archivos XML. JColibri mejora Orenge en relación con dos aspectos. Primero, la incorporación de conexión DL (Descripción Lógica). Y en segundo lugar, con respecto a la disponibilidad, Orenge es un sistema comercial y no se encuentra disponible de forma gratuita. JColibri esta a disposición de la comunidad CBR como un proyecto de código abierto.

Cabe mencionar además que JColibri proporciona una bien definida y utilizable aplicación de sistemas CBR con conocimiento intensivo (KI-CBR) a través de ontologías. Lo que

favorece el desarrollo de aplicaciones CBR basada en ontologías. El apoyo de la ontología JColibri se construye alrededor de la biblioteca OntoBridge que fue desarrollado por el grupo de investigación GAIA para administrar fácilmente las ontologías y DL (Descripción Lógica) reasoners. Se basa en el Framework Jena para el desarrollo de aplicaciones Web Semántica y hace posible la conexión con varios DL reasoners.

JColibri también incluye nuevas características para mejorar el desarrollo de sistemas CBR. La extensión de la interfaz de Web incluye varios métodos para poner en marcha un servidor Tomcat y comunicar los datos entre los procesos JColibri y las aplicaciones Web que se ejecutan en el servidor. El módulo de test de evaluación de eficiencia de las aplicaciones CBR utiliza los típicos algoritmos de evaluación N-fold (validación cruzada), Hold-Out (partición estratificada o balanceada) o LeaveOneOut (caso particular de validación cruzada). Esta extensión ha sido diseñada para ser ampliada por los usuarios que quieran crear sus propios algoritmos de evaluación, ofrece instalaciones para el almacenamiento de resultados parciales y muestra gráficamente los resultados.

Otra justificación de nuestra elección está relacionada con el hecho de que JColibri brinda la oportunidad de aplicar CBR organizada en forma de Red (CRN) modelo como estructura de Organización de la Base de Casos.

Además, JColibri proporciona una herramienta que permitirá construir aplicaciones CBR concretas (instanciar el armazón) de forma visual y guiada. Este tipo de herramientas alivian la curva de aprendizaje inicial a la que se suele enfrentar cualquiera que desee aprender a utilizar un armazón. En una primera fase nos permite obtener resultados sin tener que recurrir a la programación directa del sistema experto. Para trabajos Futuros la adaptación a nuestro sistema particular hará necesario, el ajuste y/o inclusión de código que personalice la aplicación a las necesidades del desarrollo. En este sentido ofrece también a los desarrolladores, una herramienta abierta en Java que permite incluir características particulares en JColibri para aplicaciones independientes y autónomas de sistemas de razonamiento basados en casos.

Finalmente la motivación para la elección del Framework jColibri se basa en el análisis comparativo realizado entre el Framework JColibri y los otros Framework CBR: CBR * Tools y Orenge. Por lo cual se puede determinar que el Framework JColibri mejora los otros Framework CBR: CBR * Tools y Orenge en varios aspectos: la disponibilidad (Framework de código libre), la aplicación (la implementación Java implica una gran usabilidad, extensibilidad y aceptación del usuario), interfaz gráfica (las herramientas graficas previstas facilitan el diseño del sistema).

3.2 Casos de Estudio

A continuación expondremos 3 casos en los que se desarrolla la solución a problemas similares, con diferentes métodos y herramientas.

3.2.1 Caso I: Entorno Inteligente para Prácticas Médicas en Medicina Tradicional Africana

3.2.1.1 Descripción del Caso I

Hablar de la Medicina Tradicional (TM) es hablar sobre la forma en que una comunidad hace uso de la variedad de medios para construir un sistema de atención de salud. En este sentido, existe una diversidad de TM y no hay forma de una generalización, ya que las sociedades tienen una gran diferencia en opinión, interpretación y términos terapéuticos.

El ámbito de aplicación de una sociedad en la que el término se utiliza TM tiene las siguientes características:

- No escrita, tradicional, animistas, sin una verdadera jerarquía política y el sustrato de la situación real;
- Cuando las instituciones que tienen a cargo el cuidado de las enfermedades están informados por grupos de otros ámbitos de la vida social (en forma de culto).

La Medicina Tradicional Africana (ATM) es una medicina con una larga tradición, la cual encontró su forma de servir a la gente, pero lamentablemente su práctica no es muy conocida ni comprendida. Esta medicina está estrictamente ligada a la cultura del pueblo, que considera una afección como una enfermedad social y trata de curar ambas partes visibles e invisibles existentes en la persona humana. ATM se caracteriza por el gran número de actores involucrados (paciente, profesional tradicional, fetichista, clarividente, brujo, adivino, conocido del paciente) pero también por el lugar de los rituales y plantas de en la fase de tratamientos.

3.2.1.2 Solución

El entorno inteligente se describe a continuación proporcionando un conjunto de servicios de apoyo a la práctica de la ATM, teniendo en cuenta las peculiaridades de los diferentes actores. Primero, se requiere la captura de los conocimientos de ATM, como base para la construcción de los distintos servicios para los profesionales. Como la mayoría de esas prácticas proceden de la experiencia sobre el terreno, hemos consideró que la utilización de un sistema de razonamiento basado en casos (CBR) para obtener esta información y adaptarlo a las situaciones apropiadas.

El conocimiento también es gestionado por medio de una ontología para ATM, lo que facilita la correspondencia de la terminología de ATM. En la parte principal de estas herramientas, los agentes especializados proveerán servicios especializados para los profesionales y pacientes. Este enfoque tiene varias ventajas. La primera es la capacidad de crear nuevos servicios en el sistema de forma incremental, mediante la mejora de la capacidad de agente o mediante la inclusión de nuevos tipos de agente en el sistema. Además, la asignación de un agente a cada usuario final facilitando la

personalización y adaptación de los servicios. Además, es posible considerar la colaboración entre agentes para compartir su experiencia.

Se sabe que la medicina es un complejo modelo de dominio, porque los médicos dan enfoques diferentes en un caso. CBR se ha aplicado a muchas aplicaciones médicas: en Sistemas de Diagnóstico y Apoyo en Toma de Decisiones, en Sistemas de Clasificación, en Sistemas de planificación y Sistemas de Tutoría. No hay desarrollos realizados, hasta donde se sabe, en relación a la medicina tradicional. Junto a este hecho, de acuerdo con la adaptación sigue siendo un problema de CBR en medicina.

Para este caso se utilizara un enfoque del CBR colaborativo el cual implica varios requisitos sobre los agentes. Primero, su arquitectura debe reflejar el ciclo de vida del CBR y cómo este puede ser afectado por los casos de otros agentes. Y la colaboración requiere la definición de agentes interactuando para compartir casos, así como la definición de una ontología para facilitar su comunicación.

3.2.1.3 SADMedTra

El sistema multi-agente que apoya la ATM ha sido llamado SADMedTra.

El sistema tiene como objetivo ayudar a los mejores profesionales en el proceso de diagnóstico y tratamiento, junto con una mejor comprensión de la medicina tradicional.

Los requisitos funcionales Mínimos del Sistema SADMedTra son los siguientes:

- Ser adaptable y aprender de los actores.
- Interactuar con los pacientes y los agentes del TM.
- Dar relevantes y justificados puntos de vista acerca de una enfermedad.
- Dar diferentes indicaciones para un tratamiento basado en el caso de un paciente.
- Sugerir a cada uno de los expertos de la ATM como puede curar mejor,
- Dar la información pertinente sobre las plantas medicinales (origen, composición, uso, etc.)

La solución de colaboración CBR supone un CBR por un agente cognitivo y un caso de aplicación de los actores que colaboran juntos en un aspecto positivo en la ATM. Se sabe que el aprendizaje en el MAS se puede conseguir en diferentes niveles: individual (aprendizaje aislado) y colectivo (aprendizaje interactivo). Una manera rápida de aplicar el sistema puede considerar en primer lugar el caso aislado de aprendizaje, luego el colectivo. Pero esto requiere la arquitectura de un agente para soportar la ampliación y sus consultas de la base de casos por solicitud de otros agentes. Tenga en cuenta que esto tiene algunas implicaciones en materia de

confidencialidad para cada agente que hay que tener en cuenta cuidadosamente en el diseño.

3.2.1.3.1 Arquitectura SADMedTra

La arquitectura presentada en la Figura 3.3., tiene una adecuada estructura donde todos los agentes que colaboran para el mismo objetivo, que dan un tratamiento a un paciente en una necesidad. También tenemos los objetos que sirven como entradas para los agentes en su ciclo de procesamiento. A saber hay cuatro tipos de recursos:

- Dos CBR que actúen como memoria de los agentes CBR y ejecuten al menos la primera de las tres partes de un ciclo CBR. Estos agentes utilizarán el Framework JColibri como herramienta para completar el ciclo CBR.
- Una base de datos de las plantas tradicionales.
- Un GIS (Sistema de Información Geográfica) para describir la ubicación geográfica de las plantas y donde se las encontró.
- Ontología de las prácticas de la ATM.

Estos recursos son utilizados por los siguientes agentes de diferentes capacidades y funciones:

- User Agent (UA), Agente Usuario que representa a un paciente o un usuario del sistema. Estos recopilan toda la información sobre la consulta del usuario.
- Mediator Agent (MA), Agente Mediador que transforma el conocimiento desde/hacia el agente de usuario hacia/desde los demás agentes del sistema.
- CBR Agent (CBA), Agente CBR que comienza a ejecutarse luego de la ejecución de por lo menos las tres primeras partes del ciclo CBR y no puede tomar decisiones por sí mismo. Como apoyo para este agente se utilizara el Framework JColibri.
- Service Agent (SA), Agente de Servicio que manipula los aportes de los sistemas de información geográfica para una mejor comprensión por los demás agentes del sistema
- Medicinal Plant Record Agent (ARM), Agente Registro de Plantas Medicinales que tiene capacidades similares a las SA pero referido a la base de datos de recursos.
- Ontology Agent (OA), Agente Ontología para su procesamiento y hacer uso de las ontologías ATM.

- Nganga Agent (NGA), Agente Nganga que es un agente cognitivo que realiza las funciones de curandero en ATM. Tiene un conocimiento interno
- SoothSayer Agent (SSA), Agente Adivino también un agente cognitivo que tiene un aspecto de la adivinación y puede prescribir algunos tratamientos, dependiendo del caso o la gravedad puede decir como cualquier curandero dado que puede ser capaz de tratar un caso. También tiene un conocimiento interno recopilado de su experiencia y su patrimonio tradicional.

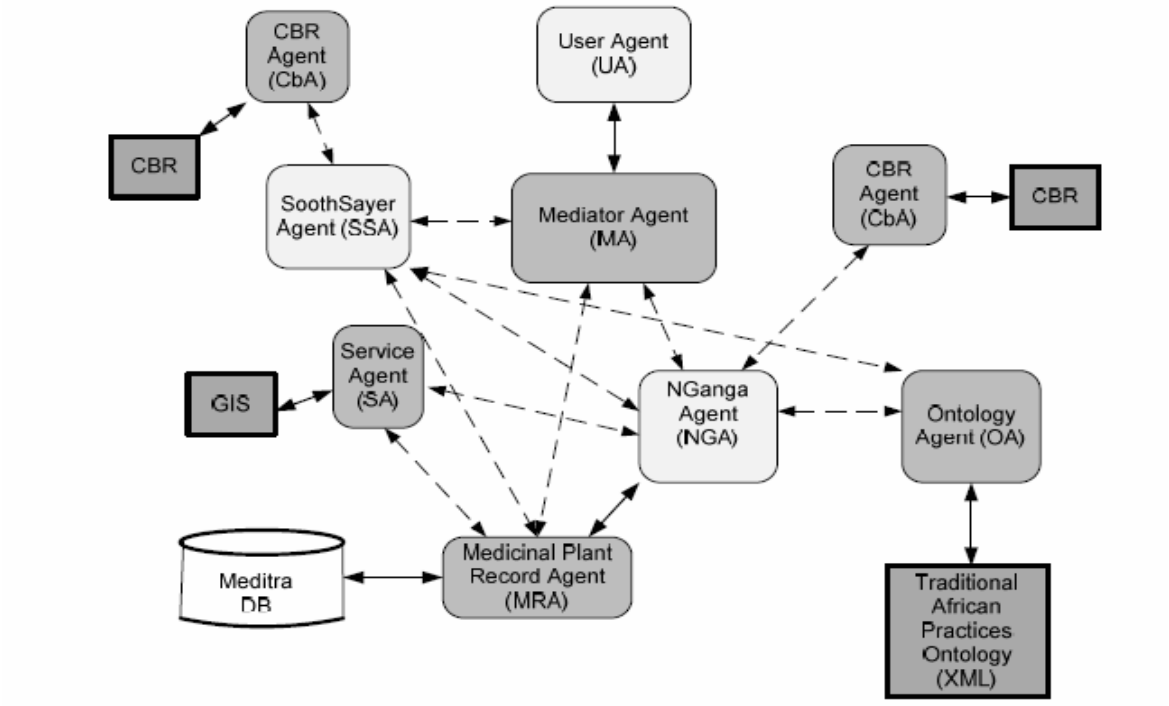


Fig. 3.4. Arquitectura del Sistema SADMedTra [46].

3.2.1.3.2 SADMedTra Ontology

Inicialmente, la ontología facilita la descripción de las interacciones de los dos agentes Agente de Usuario y Agente Nganga. Algunas de las principales entidades que se pueden señalar son los siguientes:

- Paciente: la persona que tiene un problema se refiere como una enfermedad o algún problema.
- Interventor Medtra: describe uno de los actores ATM.
- Diagnóstico: entidad relacionada con la etapa de diagnóstico.
- Síntoma: los síntomas resultantes de un diagnóstico.
- Familia: se refiere a la gran familia del paciente (padres, tíos, abuelos, etc.)
- Nación: refleja la región cultural de la concepción ATM, no como el término moderno de división de naciones.

- Aspectos Tradicionales: todos los hábitos socio-antropológicos habidos de la sociedad.
- Representación de Enfermedad: la interpretación de la enfermedad (científico y tradicional).
- Tratamiento: todos los tipos de tratamientos administrados dados por los actores ATM, con Ritual y poción incluidos.

La figura 3.4., muestra los conceptos y la interconexión de las principales entidades de la ontología en la que se basa este trabajo.

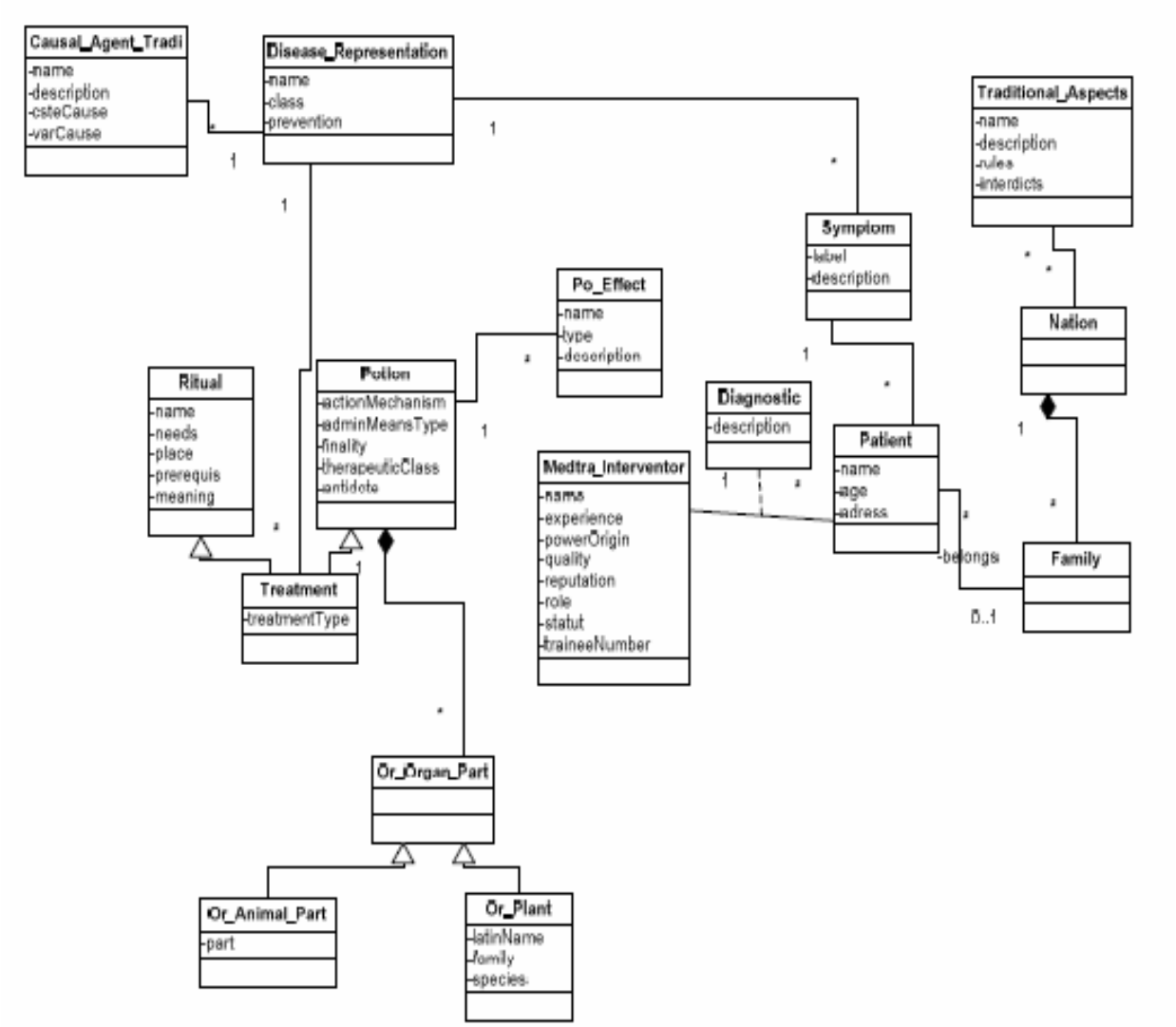


Fig. 3.5. Diagrama UML describe entidades principales de la Ontología [46].

3.2.1.3.3 Implementación CBR

La aplicación del sistema se inicia con la recopilación de información de las prácticas de ATM de las bases de conocimientos fuentes (caso de Camerún y Ruanda) con el fin de poblar la herramienta CBR JColibri que apoya a los agentes.

Los casos están estructurados con las siguientes propiedades:

- **CaseId:** Identificación de un caso.

- **Región:** Región o país para el cual el caso tiene un significado especial.
- **Síntomas:** Indicar los síntomas del caso.
- **Diagnóstico:** Nombre tradicional o local de la enfermedad o una simple descripción de cómo el profesional percibe la enfermedad.
- **Razón:** Precisa si el caso tiene una causa natural o de otro tipo. Toma el valor TRUE para un caso natural.
- **Tratamiento:** Da las ideas principales de tratar el caso en relación con el país o región del caso.

Algunos ejemplos de los casos se ilustran en la Tabla 3.1.

caseId	*region*	*symptoms*	*diagnostic*	*reason*	*treatment*
Case1	Rwanda	Lightning stricken	Calm the spirits	false	Ritual to calm the ily
Case2	Rwanda	Abortions	Ifumbi	true	Takes the Aloe sp.
Case3	Cameroon-Bamoun	A child with a big belly and who is rubbing his nose	Mansuo	false	Drink in a water l of Abu vuot, paa o
Case4	Rwanda	Cyst, migraine, buzzing one's ears	Ifumbi	false	Do sacrifice in t thers' home

Tabla 3.1. Ejemplos de Casos para ATM [46].

3.2.1.4 Análisis del Caso I

Este primer Caso presenta una solución en un entorno inteligente utilizando para ello un Modelo Colaborativo entre Agentes con CBR, se analizan los siguientes puntos.

1) Captura de Conocimiento: Se plantea que la parte principal de este caso es la captura de los conocimientos de este caso, como base para la construcción de la solución. Los casos proceden de la experiencia sobre el terreno, se ha considerado la utilización de CBR para obtener esta información y adaptarlo a las situaciones apropiadas.

2) Gestión del Conocimiento: La forma de Gestionar el conocimiento es por medio de una ontología para el caso, lo que facilita la correspondencia de la terminología del conocimiento. La Ontología definida para este caso facilita la descripción de las interacciones de los agentes y facilita la comunicación entre ellos. Compuesta por 2500 recetas tradicionales basadas en alrededor de 400 productos vegetales, minerales y animales. Esta ontología tiene una estructura jerárquica, que representan las plantas curativas de las ATM.

3) Arquitectura: Los agentes especializados proveerán servicios especializados para los usuarios en sus tipos de consultas. Esta refleja el ciclo de vida del CBR y la interacción con los agentes definidos para esta solución.

4) Herramienta Utilizada: Como herramienta de apoyo se utiliza el jColibri para asistir a los agentes CBR en sus procesos, en este caso el JColibri no es el tema principal de este caso analizado, pero se utiliza para apoyar a uno de los componentes de la Arquitectura de Agentes planteada para este caso.

5) Base de Conocimientos: Se logró una base de conocimientos estructurada de acuerdo a la aplicación del sistema para lo cual se realizó la recopilación de información de los casos de las prácticas que formarán parte de la base de conocimientos tomando fuentes validas con el fin de poblar la herramienta de CBR jColibri que apoya a los agentes.

3.2.2 Caso II: Sistema CBR en la Contraloría General de la Republica

3.2.2.1 Descripción del Caso II

Demora en la emisión de Informes de Control debido a, entre otros:

- Falta de uniformidad de criterio para la identificación de responsabilidades.
- Acceso limitado a conocimientos colectivos e información (casuística, consultas, criterios normativos, etc.), debido a la falta de sistematización y registro.
- Capacidad operativa limitada de las unidades orgánicas de accesoria para atender oportunamente las consultas formuladas.
- Demora en la búsqueda de antecedentes, debido a restricciones en su acceso.

El personal que participa en la ejecución de acciones de control, requiere contar con información legal que comprenda:

- Una base de datos ágil y amigable de casos detectados en anteriores acciones de control.
- Acceso a las opiniones legales emitidas por la CGR y pronunciamientos de otras entidades rectoras, así como a doctrina, jurisprudencia y normativa relacionada.

3.2.2.2 Solución

Implementación de un Sistema de Gestión del Conocimiento que utilice la metodología del Razonamiento Basado en Casos (Case-Based Reasoning – CBR), para la identificación de responsabilidades de tipo civil, penal y administrativo funcional. El Sistema debe estar disponible en dos versiones (on-line) y standalone, así como brindar acceso a fuentes de información vinculadas.

El Sistema consiste en una base centralizada de casos tipo, elaborados a partir de observaciones tomadas de informes de Control con responsabilidades de tipo civil, penal o administrativa funcional, los cuales muestran las condiciones, metodología y procedimientos sugeridos para la identificación de responsabilidades en las acciones de control respecto a situaciones irregulares similares a las contenidas en dichos casos. En otras palabras, en cada uno de estos casos almacenados en esta base de datos centralizada se recoge la experiencia acumulada por cada uno de los profesionales de la Contraloría General de la República en el ejercicio del control gubernamental.

La implementación del sistema de RBC en la CGR tiene por objetivo además de la unificación de criterios en la institución en lo referente a la tipificación de delitos de acuerdo con las situaciones evidenciadas; contar con información legal oportuna, ágil y confiable, de manera que permita incrementar la calidad de las consultas legales y reducir los tiempos utilizados hoy para responder estas consultas.

Esto se realizará con los objetivos de Contar con información legal oportuna, ágil y confiable para la gestión del control gubernamental; ya que por ser la razón de ser de la CGR en las acciones legales se necesita información legal de calidad, rápida y confiable que permita mejorar las consultas legales y mejorar los tiempos de respuesta de estas consultas, ya que estas se realizan de forma muy lenta, o son muy trabajosas y extensas para resolverlas en un tiempo optimo, la implementación de esta mitología lo que se desea es mejorar la forma como se realizan estas consultas y puede verse como un intento de sistematizar y poner en práctica las actividades operativas que se requieren para que se logre adecuadamente cada uno de las etapas del aprendizaje organizacional.

3.2.2.2.1 Arquitectura del Sistema CBR

La adaptación del ciclo CBR al proyecto para la CGR da como resultado el proceso esquematizado en el diagrama. En el proceso de gestión del conocimiento de la Contraloría, participan todos los funcionarios involucrados en la ejecución de las acciones de control:

- De una parte, y en una primera etapa, se clasifican, revisan e incluyen los nuevos casos en la base de casos para que puedan ser consultados por otros usuarios.

- De otro lado, posteriormente, los usuarios proporcionan al sistema las consultas y condiciones, de manera que éste recupere de la base de casos los que más se asemejen al actual. El usuario que consulta, revisa la solución sugerida por el sistema, la adapta a las circunstancias o la completa para resolver el caso presente.

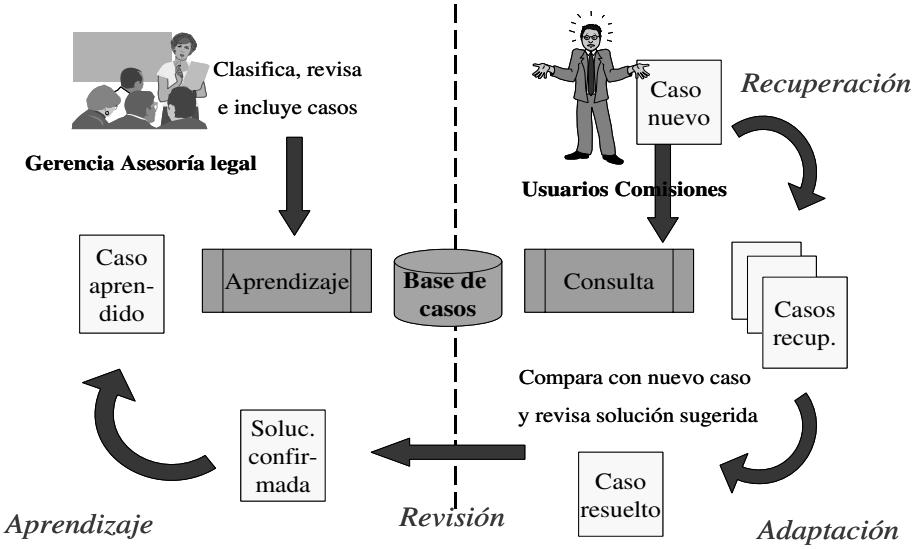


Fig. 3.6. Arquitectura del Sistema CBR en CGR [44].

La solución confirmada será la que se utilice para resolver el caso. Esta solución puede constituirse en un nuevo caso e incorporarse en la base de casos única. También puede incorporarse la retroalimentación obtenida del seguimiento de las recomendaciones o de la decisión que tome el poder judicial para enriquecer la base de casos.

3.2.2.2.2 Metodología propuesta

- **Para la creación de la base de casos**

Se plantea entonces, empezar estructurando y categorizando los casos, de manera que puedan crearse vistas de acuerdo con indicadores definidos, para luego hacer sobre esta base de casos una consulta tipo base de datos documental. Como ejemplo se estructura el caso de malversación de fondos en la figura 3.6.

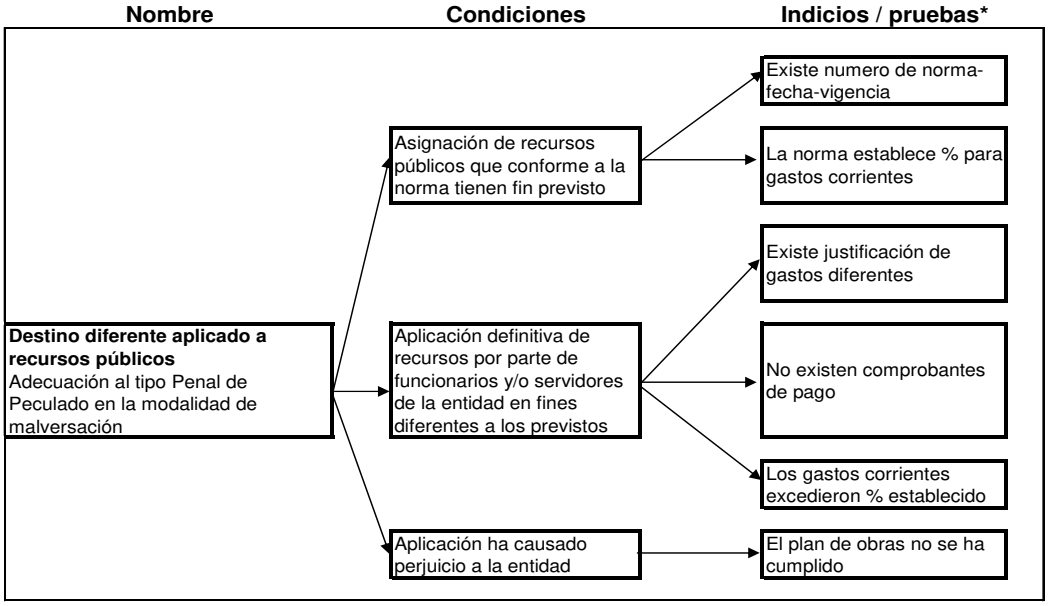


Fig. 3.7. Representación del Caso Malversación de Fondos [44].

Esto permitiría organizar el conocimiento para en una etapa posterior adquirir o desarrollar el motor de inferencia necesario que incluya las funciones de recuperación, reutilización, revisión y recuerdo que caracterizan el CBR.

- **Para la incorporación de funciones CBR**

La base de casos creada podrá administrarse inicialmente con un sistema de gestión documental cuyo principal objetivo será almacenar información textual contenida en documentos, tales como las normas jurídicas, investigaciones, libros, imágenes, informes y acciones de control que puedan relacionarse recíprocamente permitiendo búsquedas o recuperaciones por texto completo o por descriptores.

Los casos recopilados por el área legal serán estructurados de acuerdo con las condiciones necesarias para que se configure y las situaciones que deben evidenciarse para que se cumpla cada condición.

Estos casos deben ser analizados y calificados cada vez que se presenten antes de ser actualizados en el sistema, debido a que se quiere lograr el objetivo de tener la información como la opinión institucional

Cada caso estará identificado con un nombre, una adecuación y un tipo que puede ser civil, penal o administrativo. El nombre y la adecuación del caso, estarán ligados a su vez a la ley, reglamentación o norma que lo define y a la doctrina que se encuentre al respecto, lo que constituye la base de conocimiento general.

Cada uno de los casos de la base estará documentado con el número de informe de control, consulta o denuncia en el que se encuentra y a la vez de la acción o actividad de control de la cual forma parte (esta última información podrá ser extraída del sistema de Auditoría Gubernamental (SAGU).

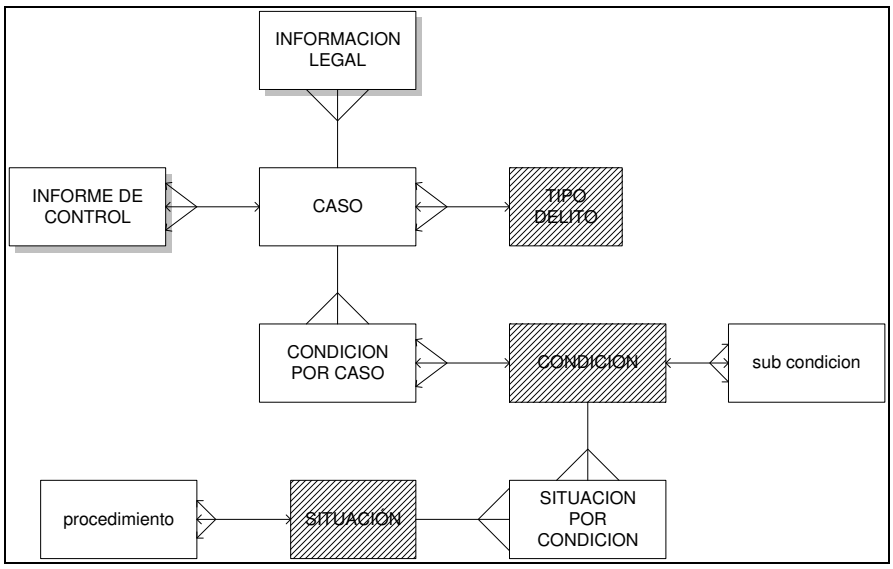


Fig. 3.8. Modelo Entidad Relación de la Base de Casos [44].

3.2.2.2.3 Funciones Principales

- **Actualización o llenado de los casos:** Módulo que permite al usuario "creador" de conocimiento, ingresar al sistema las situaciones, condiciones y factores necesarios para que se configure cada caso, así como establecer el vínculo entre el caso y la información adicional de cada uno.

- **Búsqueda de información:** Módulo que permite la búsqueda de información en la base de casos de acuerdo con parámetros definidos como palabras clave o descriptores dentro de las condiciones y situaciones. Solicita una palabra o frase a buscar y muestra en pantalla los casos que tienen esa frase en sus condiciones o situaciones.

- **Inferencia de casos:** Motor de búsqueda por situaciones que luego podrá ser mejorado o remplazado por el motor de inferencia de un CBR. Permitirá la selección por listas de una o varias condiciones para configurar y mostrar los casos que cumplan con ellas, ordenados de acuerdo a una probabilidad ponderada:

    - **Prob. Ponderada** = (Factor condición / N° situac por condición) * importancia de la situación en la condición

    - **Factor condición** = (100 / N° condic. por caso) * importancia de la condición en el caso

- **Resumen o ranking de casos:** Se trata de una consulta en la cual se mostrarán los casos mas frecuentes ordenados en forma descendente.

- **Consulta:** Consulta de doctrina, jurisprudencia, normas relacionadas y detalle documental y procesal.

Permitirá consultar información adicional relacionada con el caso que se consulta extrayéndola del sistema de información legal, el sistema de auditoria gubernamental, documentos digitalizados, etc. Que puedan formar parte de la base de datos documental.

Esta opción, podrá incluir un acceso rápido a consultas de otros aplicativos en los que se pueda profundizar la información.

3.2.2.2.4 Algunos Casos

Algunos ejemplos de los casos se ilustran a continuación.

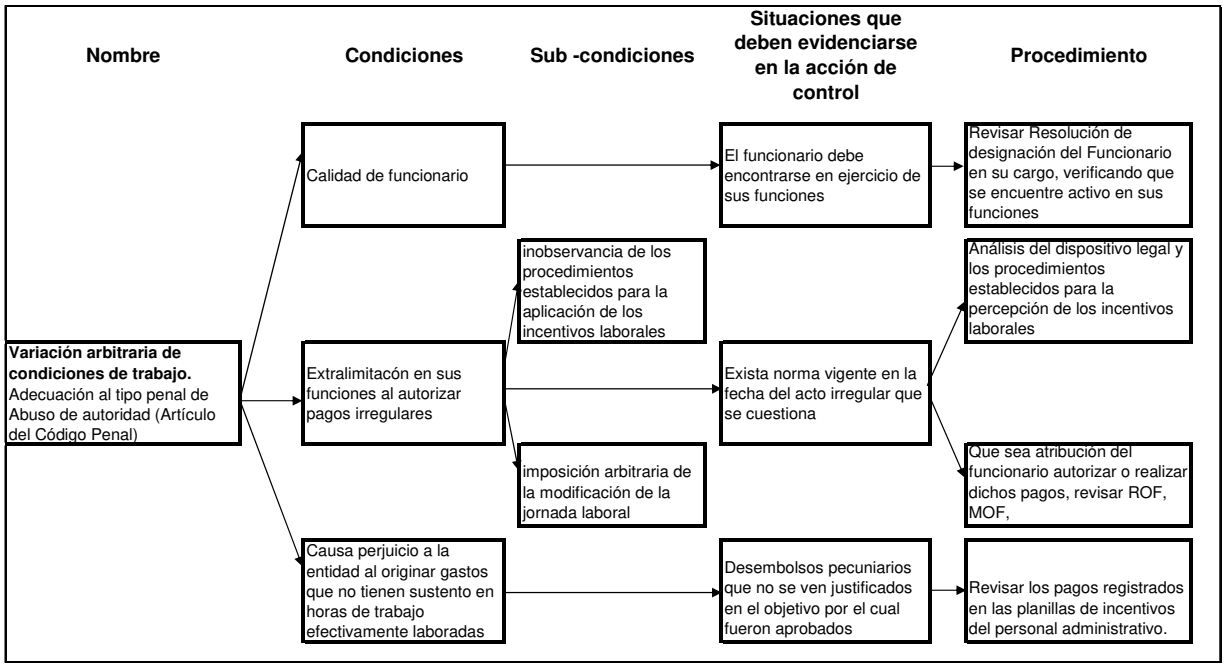


Fig. 3.9. Representación del Caso – Abuso de Autoridad [44].

3.2.2.2.5 Resultados

Con la Implementación del sistema se tienen las siguientes metas alcanzadas. Sistema CBR instalado en servidor de la CGR implementado con 19 estructuras (Civil, Penal –17 y Administrativa) y 67 casos tipo, disponible en versión intranet y standalone, el Sistema tiene 3 tipos de búsqueda (navegación, avanzada y guiada), permite la interacción con los usuarios (comentarios generales y específicos), permite el acceso a informes de control, citas de jurisprudencia, doctrina, normas legales, situación judicial, direcciones Web y recomendaciones generales.

3.2.2.3 Análisis del Caso II

Este Segundo Caso presenta como solución un Sistema de Gestión del Conocimiento que utiliza la técnica de CBR, se analizan los siguientes puntos.

1) Captura de Conocimiento: La captura de los conocimientos de este caso, se recoge la experiencia acumulada por cada uno de los profesionales de la Contraloría General de la República en el ejercicio del control gubernamental.

2) Gestión del Conocimiento: La forma de Gestionar el conocimiento es por medio de una metodología que plantea, empezar estructurando y categorizando los casos, de manera que puedan crearse vistas de acuerdo con indicadores definidos. Los casos recopilados serán estructurados de acuerdo con las condiciones necesarias para que se configure y las situaciones que deben evidenciarse para que se cumpla cada condición. Estos casos deben ser analizados y calificados cada vez que se presenten antes de ser actualizados en el sistema.

3) Arquitectura: La Arquitectura de este caso presenta dos Bloques definidos uno donde se Ingresan los Casos a considerar por los expertos y otra donde se consulta

algún caso por los usuarios, encontrando una solución o amoldando la semejante para el caso consultado.

4) Base de Conocimientos: Formada de casos tipo, elaborados a partir de observaciones tomadas de informes almacenados recogidos de la experiencia acumulada por cada uno de los profesionales expertos en el tema.

3.2.3 Caso III: JaDaCook: Java Application Developed and Cooked Over Ontological Knowledge

3.2.3.1 Descripción del Caso III

JaDACook es un sistema de Razonamiento Basado en Casos aplicado al dominio de la gastronomía. Este sistema ha sido desarrollado en java usando jColibri, para participar en el Concurso de Cocina Computarizada que se celebró en la Conferencia Europea sobre Razonamiento Basado en Casos 2008. El sistema es también la evaluación final de un curso de postgrado de Enseñanza Computarizada de la Facultad de Ciencias de Computación (Universidad Complutense de Madrid).

La aplicación presenta la funcionalidad para resolver tres pruebas propuestas por el Concurso de Cocina Computarizada, estas pruebas son las siguientes:

1) **Prueba del plato simple:** Dada una consulta, el resultado será la recuperación y adaptación de una receta para un único plato.

2) **Prueba de la negación:** Se deben ingresar consultas con los ingredientes que el usuario no desea que aparezca en la recomendación final por motivos personales.

3) **Prueba del menú:** Se debe ingresar un plato simple, y el sistema recomendará dos platos más para completar el menú, según el conocimiento adquirido por la primera prueba.

3.2.3.2 Solución

A continuación se presenta una breve descripción de cómo la aplicación lleva a cabo los principales procesos del ciclo CBR.

3.2.3.2.1 Adquisición de conocimiento

Este proceso se ha distribuido en dos fases. En primer lugar, se examinan los recursos existentes para la reutilización, y en segundo lugar, se formaliza los conocimientos adquiridos. Primero se recopila información sobre los tipos de ingredientes, cocción y tipos de dietas de diversas fuentes, como los expertos, los diccionarios y las páginas Web de cocina.

Los conocimientos adquiridos se han estructurado, conceptualizado y formalizado en OWL como ontología reutilizable. Además, hay un proceso de transformación de las recetas con el fin de trabajar con ellos de una manera mejor. Ambos puntos se describen en las siguientes subsecciones.

- **Ontología**

La ontología creada para esta aplicación incluye más de 200 ingredientes para cubrir la mayoría de recetas proporcionadas por la organización del concurso y para implementarla se utilizó la herramienta denominada Protege.

Esta ontología tiene una estructura jerárquica, siendo la receta la raíz y los ingredientes sus hijos directos. Presenta 206 clases y estas se dividen en diferentes subclases que agrupan los ingredientes de la mejor manera posible para el logro del objetivo:

- **Origen animal.** Ingredientes de origen animal. Se agrupan en las categorías pescado, carne, leche, queso y huevos. El subárbol se muestra en la Figura 2 (que muestra todas las subclases).
- **Origen vegetal.** Ingredientes de origen vegetal. En este caso, cereales, especias, frutos secos, frutas y verduras son las clases más comunes.
- Edulcorantes, Bebidas y básicos.

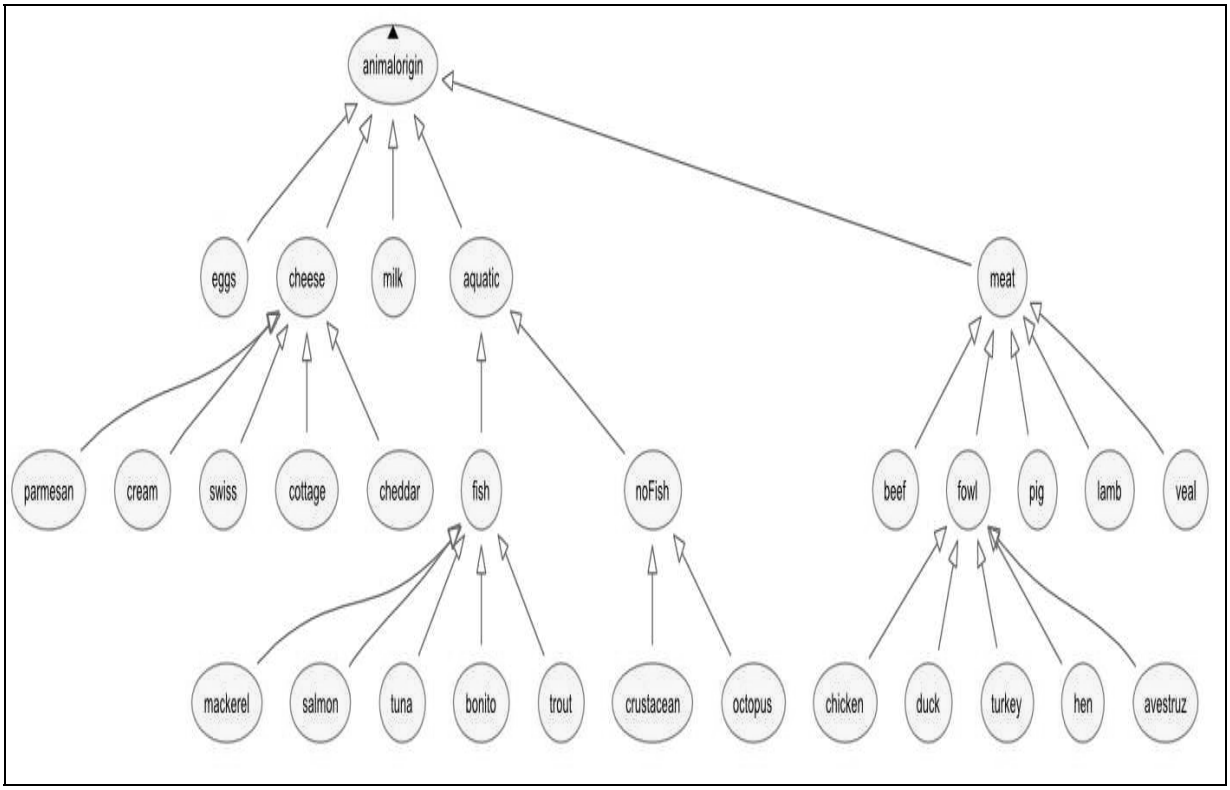


Fig. 3.10. Subárbol de la categoría Origen animal [45].

A continuación se muestra el árbol jerárquico de la ontología desde la aplicación.

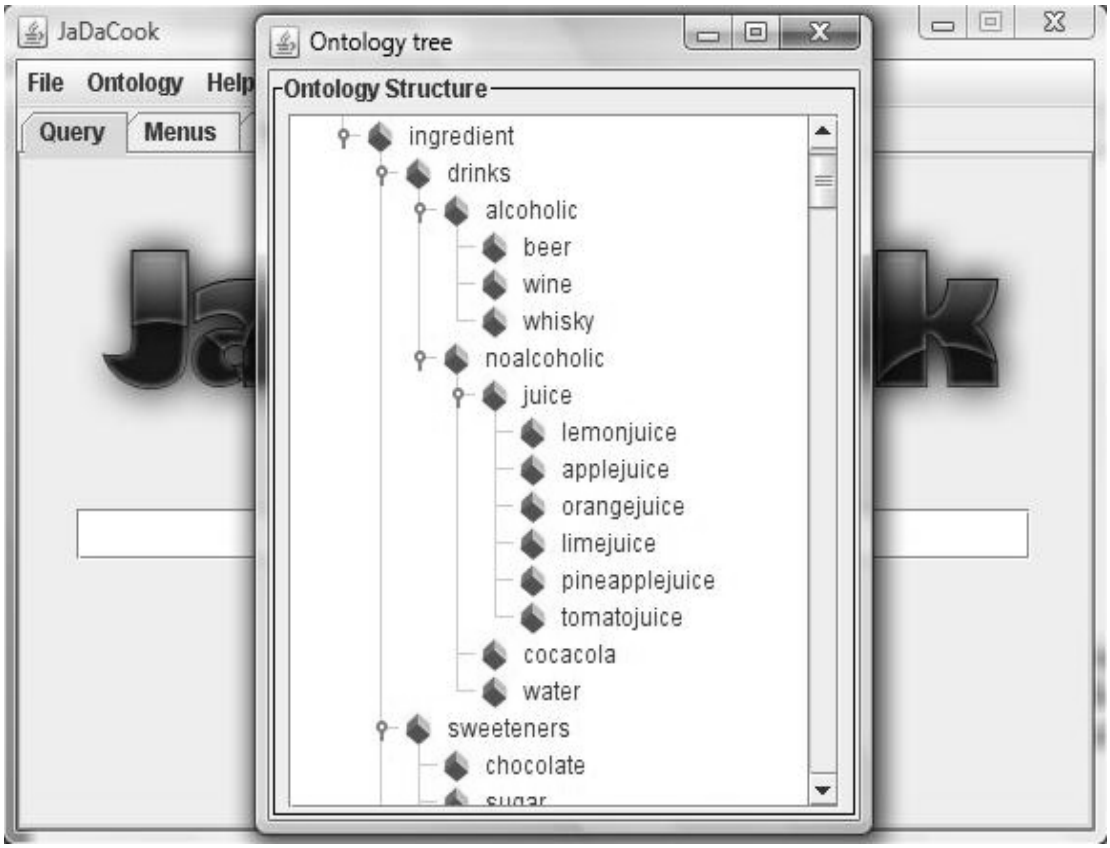


Fig. 3.11. Estructura Jerárquica de la Ontología [45].

- **Casos**

Las recetas proporcionadas por la organización del concurso se encuentran en un archivo XML bien organizado, pero a pesar de esto, para este trabajo la información más relevante fue extraída y agrupada en casos para su posterior análisis. Usamos la estructura de un caso con los siguientes atributos: Número de caso, el título de la receta, lista de ingredientes y el proceso de desarrollo (descripción textual). Con el fin de realizar una búsqueda eficaz y obtener los mejores resultados, se hizo necesario vincular cada caso de la base de recetas con las clases correspondientes de la ontología. De esta manera, el mapeo que surgió fue sencillo, cada ingrediente es del tipo de la clase ingrediente y el número de casos es dado por la clase raíz receta.

3.2.3.2.2 Recuperación

El proceso de recuperación consiste en obtener los casos más similares a una determinada consulta. Se han concentrado esfuerzos en la búsqueda de una buena medida de la similitud, que reúna los más similares entre sí, y de un método para encontrar la mayoría de los casos apropiados. El método de recuperación que se utilizó es el vecino más cercano, que es el más usado para comparar los atributos de los casos en forma numérica. Se hace uso de funciones globales para comparar los casos y las funciones de similitud locales para comparar los atributos de los casos. Se realiza una búsqueda secuencial de todos los casos mediante la aplicación de las funciones descritas posteriormente, y se selecciona las descripciones del evento que mejor se ajusten a la petición.

- **Similitud**

Se incluye dos funciones de similitud para comparar los títulos y los ingredientes de la consulta y el caso. Sin entrar en detalles de la aplicación, a continuación se presentan algunas de sus características:

- **ContainsTittle.** Esta función se encarga de comparar si el título de una receta es igual o no a la consulta dada. En concreto el resultado del cálculo será uno si ambas partes son iguales, o cero de otro modo. Se utiliza principalmente para los diferentes tipos de recetas.

- **ContainsIngredients.** Esta función tiene por objetivo determinar si los ingredientes a recuperar están en el caso actual y cuántos de ellos están en el, asignando una valor entre 0 y 1, dependiendo del número de ingredientes y si estos son exactamente los que el usuario está buscando. Hay dos posibles alternativas para hacer frente a esta situación:

  El ingrediente X se encuentra en la descripción del caso. La similitud de este ingrediente será 1, por lo que se agrega este valor al cálculo.

  El ingrediente X no se encuentra en la descripción del caso. Hay dos posibles acciones para determinar si hay algunas similitudes. Si ese ingrediente no es un último nodo del árbol de la ontología. En este caso, habrá que buscar los hijos de este ingrediente (padre), este caso es descrito con mayor detalle posteriormente. Si el ingrediente es nodo final del árbol de la ontología, si no se puede especificar más este ingrediente, la similitud se similitud será buscada en sus nodos hermanos. Este proceso de reutilización se explica en una sección posterior.

Una vez que todos los ingredientes han sido procesados, el valor de cálculo será normalizado al rango [0,1]. Finalmente, las K recetas más similares se mostrarán al usuario.

- **Cálculo de hijos**

Como se explica en la sección anterior, en el supuesto de que uno de los ingredientes solicitado por el usuario es demasiado genérico, se especifica utilizando la ontología creada para este trabajo. Para ello, se utiliza la biblioteca OntoBridge, escrita en Java, que proporciona el administrador de ontologías. La tarea es realmente simple, lo único que hay que hacer es seguir el camino de la clase del ingrediente genérico hasta llegar a las hojas del árbol de la base de conocimiento. La similitud en este caso, será uno si hay un hijo que corresponda con la descripción de la receta y cero en caso contrario.

3.2.3.2.3 Reutilizar y Retener

La etapa de reutilización es responsable de adaptar la solución de los casos recuperados para los requisitos de las consultas, si fuera necesario. En el presente sistema, la estrategia de reutilización empleada se llama Reinstantiation y es una especie de estrategia basada en la sustitución, dicha sustitución tiene lugar entre los elementos con la misma función pero con diferentes valores. Por lo tanto, el proceso de adaptación es básicamente la sustitución de los ingredientes que no han sido pedidos por la consulta proporcionando para ello una similitud entre hermanos, una vez más utilizando la ontología creada para este fin.

Es importante señalar que la similitud en el caso de una adaptación no es total, porque como su nombre sugiere adaptar (sustituir), con cualquier ingrediente similar, y es por eso que la similitud en este caso es del 0,9 al adaptar el ingrediente. Por último, un aspecto que no se ha discutido es que la adaptación se realiza en el ámbito de la aplicación de los hermanos, una decisión basada en el hecho de que si también son adaptados por primos y otros parientes del árbol de la ontología, la solución a la pregunta sería muy lejano a los ingredientes necesarios, dando resultados indeseados. La fase de aprendizaje (retener) tiene como función principal el almacenamiento de los casos en que han sido adaptados, en la base de conocimiento. Por lo tanto, estos nuevos casos se puede utilizar en el futuro para las búsquedas y así la recuperación será más fiable. En este sistema, se deja al usuario la decisión de que casos almacenar, de esta forma se almacenarán los casos más importantes que tienen la mejor solución al problema.

3.2.3.2.4 Resultados

En esta sección se da una descripción general sobre los resultados preliminares y dictámenes de usuarios externos que han señalado el camino más fácil para interactuar con la aplicación.

Resultados de la prueba del plato simple

P1: Cocinar, un plato principal con carne y coliflor. Los resultados se dan en la siguiente figura.

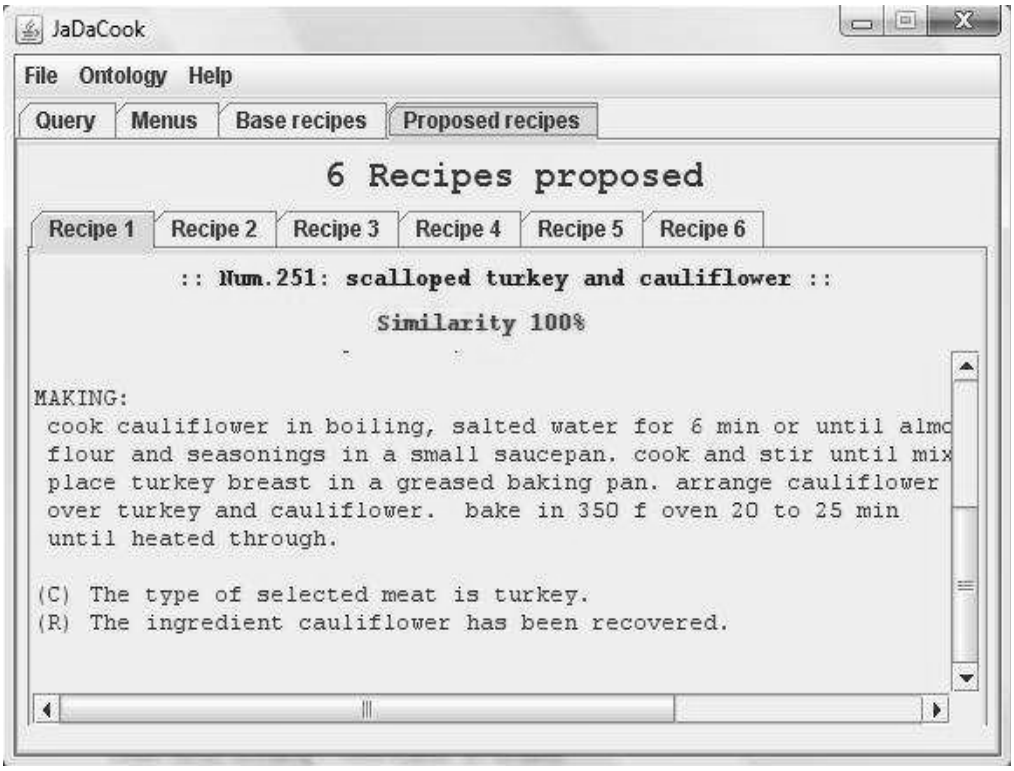


Fig. 3.12. Resultado de la Primera Prueba con Similitud 1 [45].

Hubo 3 recetas encontradas con similitud de valor 1, y 3 recetas con similitud de 0,9. Como se muestra en la siguiente figura, la coliflor ha sido sustituida por el brócoli.

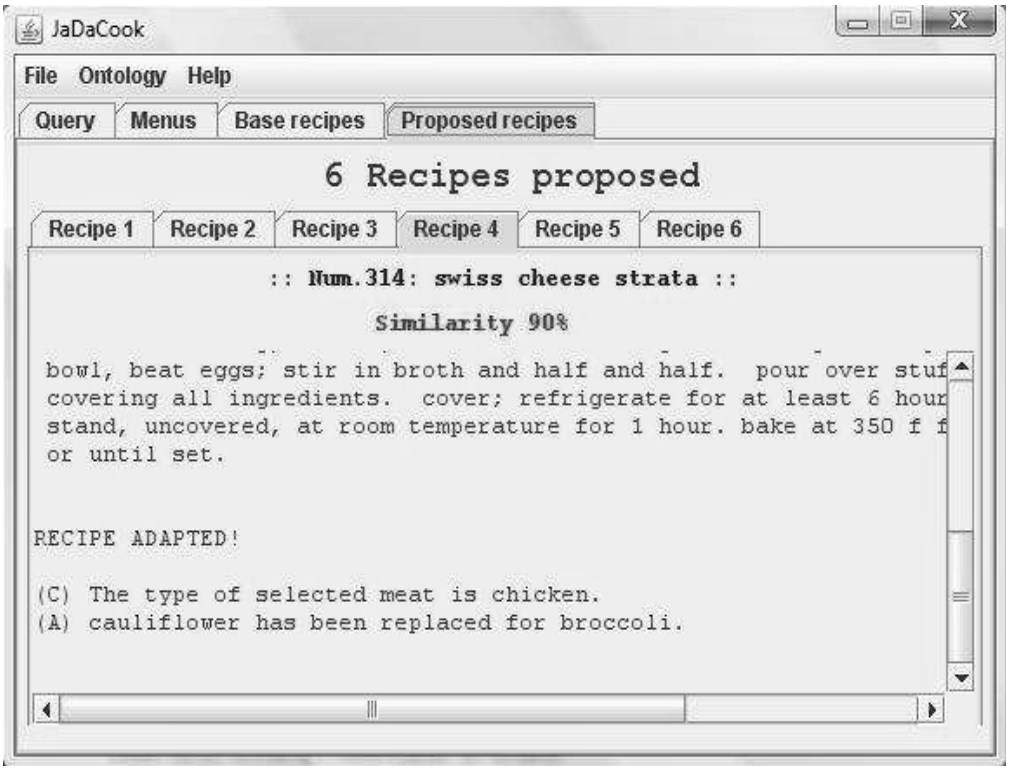


Fig. 3.13. Resultado de la Primera Prueba con Similitud 0,9 [45].

3.2.3.3 Análisis del Caso III

Este último Caso presenta un sistema CBR desarrollado con jColibri, cuya principal característica se basa en una buena definición de la ontología utilizada, la cual se basa en una estructura jerárquica de las recetas (raíz principal) y sus ingredientes (nodos hijos), se analizan los siguientes puntos.

1) Captura de Conocimiento: dentro de este caso primero se recopila información sobre el conocimiento a tomar en cuenta para lo cual este proceso se ha distribuido en dos fases. Primero, se examinan los recursos existentes para la reutilización, y Luego, se formaliza los conocimientos adquiridos

2) Gestión del Conocimiento: La forma de Gestionar el conocimiento es por medio de una ontología para el caso, la ontología creada para esta aplicación incluye más de 200 ingredientes para cubrir la mayoría de recetas proporcionadas por la organización del concurso y para implementarla se utilizó la herramienta denominada Protege. Esta ontología tiene una estructura jerárquica, siendo la receta la raíz y los ingredientes sus hijos directos. Presenta 206 clases y estas se dividen en diferentes subclases que agrupan los ingredientes de la mejor manera posible.

3) Arquitectura: Presenta una Arquitectura que refleja el ciclo de vida del CBR recuperar, reutilizar, revisar y retener, para lo cual se apoya en el uso de la herramienta jColibri que contiene estos procesos en su estructura.

4) Base de Conocimientos: Se logró una base de conocimientos estructurada y se vinculó cada caso de la base de recetas con las clases correspondientes de la ontología. De esta manera, el mapeo que surgió fue sencillo, cada ingrediente es del tipo de la clase ingrediente y el número de casos es dado por la clase raíz receta, con el fin de realizar una búsqueda eficaz y obtener los mejores resultados.

5) Herramienta Utilizada: Como herramienta de apoyo se utiliza el jColibri para crear la solución para este caso presentado.

CAPITULO IV

IMPLEMENTACIÓN TEÓRICA

4.1 Características de la Institución: HSP S.A.C.

En este apartado, se describe a la institución donde se aplicará el caso práctico de este trabajo. Se da a conocer cuál es el giro de la empresa, así como la metodología que utiliza para desarrollar su principal actividad de negocio.

4.1.1 Descripción

Empresa que brinda consultoría en Tecnología de Información. Como Partner de Oracle se especializa en la implementación del ERP E-Business Suite, así como personalizaciones y localizaciones a medida del cliente. La experiencia y calificación de sus consultores permite realizar un análisis del nivel de tecnología implantado en sus clientes y recomendarle las mejores soluciones, necesarias para el éxito de su negocio con un alto valor agregado y con visión de futuro.

4.1.2 Misión

Ser la mejor compañía experta en el uso de tecnología abierta de la información.

4.1.3 Visión

Lograr que sus soluciones sean una fuente de ventajas competitiva y continua para sus Clientes.

4.1.4 Valores

- **Compromiso.** Política "Cliente-Socio" para generar una relación con los clientes, que va más allá de los compromisos contractuales y que se implementa mediante el entendimiento de los objetivos comunes de cada proyecto. Esta política se inicia, estableciendo una relación de colaboración y compromiso entre todos los ejecutivos involucrados, para cumplir los objetivos propuestos.

- **Trabajo en Equipo.** Desarrollo de modelos de negocios creativos permitiendo que la organización del cliente sea más rentable y productiva.

- **Integridad.** Comprometidos con una vocación de servicio que se sustenta en una actitud profesional con la transparencias en su comunicación y con la seguridad del manejo de su información.

- **Sentido de Urgencia.** Se traduce en cada una de las actividades que realizan las personas que laboran en la consultora con el fin de cumplir la estrategia de confiabilidad antes del plazo, no dejando para mañana lo que pueden hacer hoy.

4.1.5 Metodología de Implantación

La metodología de implantación utilizada por la organización es Oracle AIM (Applications Implementation Method) probada para implementaciones de aplicaciones de clase mundial con herramientas y procedimientos agrupados por fases, que guían al cliente y consultor por un camino probado, seguro y eficaz. Ayudan al cumplimiento en tiempo y estructura del Proyecto [5].

AIM usa fases de proyecto para incluir calidad y puntos de control y permitir la coordinación de las actividades de proyecto a todo lo largo de la implementación. Durante las fases, el equipo del proyecto ejecutará tareas en varios procesos [5].

En la siguiente figura se muestra la relación entre las fases y los procesos [5].

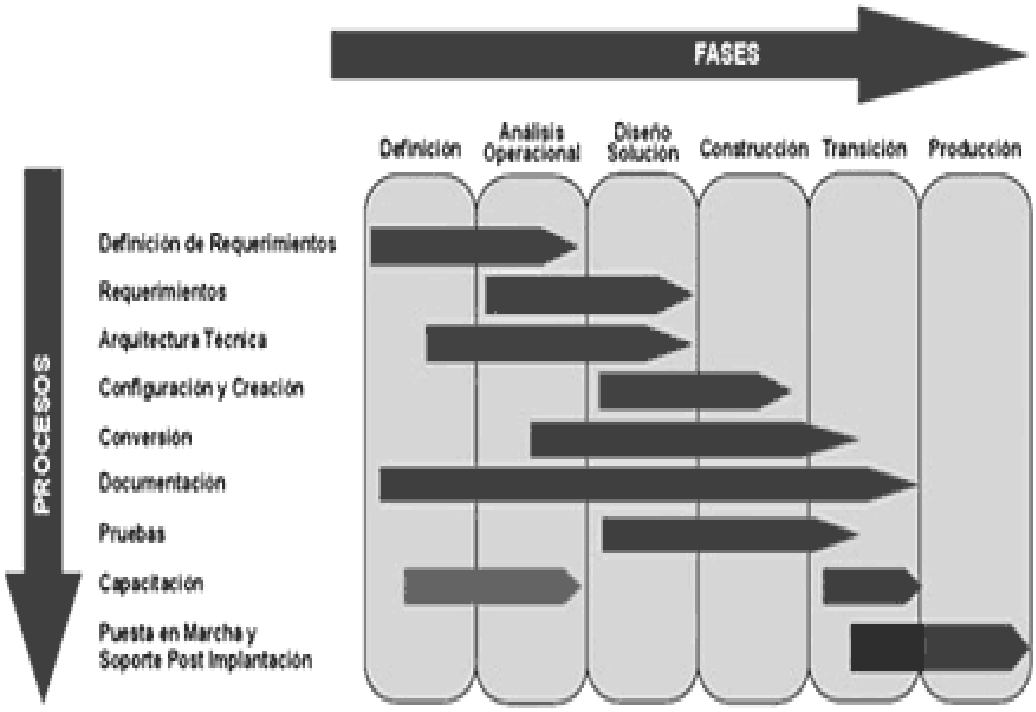


Fig. 4.1. Metodología AIM [5].

A continuación, se da una pequeña descripción para las fases y procesos.

4.1.5.1 Fases de Proyecto

Las fases de proyecto son las siguientes [5]:

4.1.5.1.1 Definición

En esta fase, el equipo de gestión de proyecto planea el proyecto de implantación. Los objetivos son identificar requisitos de negocio y de sistema, proponer el modelo comercial futuro, y proponer la aplicación y la arquitectura de tecnología de la información. El equipo desarrolla un plan educativo para asegurar que los miembros del equipo reciban entrenamiento y soporte necesario para realizar sus papeles en el proyecto.

4.1.5.1.2 Análisis Operacional

Durante la fase de Análisis Operacional, el equipo de proyecto recoge información sobre los procesos de negocio del usuario final y sus requerimientos. El equipo de proyecto desarrolla escenarios de los requerimientos del usuario para evaluar el nivel ideal entre los requisitos comerciales detallados y la funcionalidad estándar de aplicación. También se crea un modelo para la estructura de la aplicación y se sugiere una arquitectura técnica.

4.1.5.1.3 Diseño Solución

El propósito de esta fase es crear la solución de procesos de negocio óptima para encontrar los requerimientos futuros. Durante esta fase, los miembros del equipo de proyecto diseñan opciones de configuración de la aplicación y documentan detalladamente los procesos de negocio. El diseño detallado de cualquier extensión, personalización, interfase y conversión de datos ocurren durante esta fase. El equipo también identifica cambios organizativos requeridos para la implementación.

4.1.5.1.4 Construcción

Durante la fase de construcción, el equipo de desarrollo codifica y prueba todas las extensiones personalizadas incluyendo mejoras de la aplicación, conversiones, e interfaces. El equipo crea y ejecuta test scripts de desempeño, integración y funcionalidad del sistema.

4.1.5.1.5 Transición

Durante la transición, el equipo de proyecto despliega la aplicación terminada en la organización. La transición depende de la fase de construcción para que el sistema sea completamente probado. El equipo de proyecto ejecuta la conversión de datos y usa la documentación desarrollada para entrenar a los usuarios finales y al equipo de mantenimiento.

4.1.5.1.6 Producción

La fase de producción es la última fase de la implementación y el comienzo del ciclo de soporte del sistema. El curso de esta fase provee al cliente el soporte necesario para el resto de vida del sistema.

4.1.5.2 Procesos

Los procesos de proyecto son los siguientes [5]:

4.1.5.2.1 Definición de Requerimientos

Se define las necesidades de negocio que deben ser resueltos para la implementación exitosa de la aplicación.

4.1.5.2.2 Arquitectura Técnica

El equipo de proyecto diseña una arquitectura de sistemas alrededor de la visión comercial de la organización. Este diseño debe incluir la aplicación Oracle, aplicaciones de terceros y aplicaciones personalizadas; hardware computacional; y las redes y la infraestructura de comunicaciones de datos.

4.1.5.2.3 Configuración y creación

Este proceso produce extensiones personalizadas de la aplicación para satisfacer los requerimientos identificados durante el proceso de Identificación de Requerimientos. Las extensiones personalizadas incluyen módulos de programa que deben ser diseñados, construidos, y probados antes de que puedan ser incorporados a la aplicación.

4.1.5.2.4 Conversión

Este proceso define las tareas para la carga inicial a las tablas de Oracle Application. El primer paso de este proceso es definir los datos necesarios provenientes de otros sistemas, los cuales serán importados a la nueva aplicación. Luego se procederá a la carga, pruebas y validación de la funcionalidad del sistema con la información importada.

4.1.5.2.5 Documentación

La documentación de los requerimientos e implantación están estrechamente correlacionados, y la cantidad y nivel de detalle de la documentación varían por proyecto.

4.1.5.2.6 Pruebas

Este proceso integrado busca probar la calidad de todos los elementos de la aplicación.

4.1.5.2.7 Puesta en marcha

El objetivo de este proceso es emigrar la organización, los sistemas, y empleados al nuevo sistema. Los objetivos adicionales incluyen monitorear y afinar el sistema de producción. Este proceso abarca el pase a producción, y el soporte de postproducción.

4.2 Planeamiento de la Solución del Problema

El planteamiento propuesto para la Solución del Problema se basa en el análisis realizado para los casos de estudio descrito en el Capitulo III, de los cuales se ha identificado 5 puntos importantes a considerar para la solución de este tipo de herramientas de Razonamiento Basado en Casos (CBR) por lo cual se ha considerado aplicar el mismo modelo de análisis hecho en los casos de estudio por considerarlos como los mas importantes para realizar la solución del problema, y es en base a estas consideraciones que se ha realizado el presente planteamiento de la solución del problema.

Además considerando que el proyecto tiene como objetivos principales, contar con información oportuna, ágil y confiable para la gestión del conocimiento y la toma de decisiones en proyectos de Implementación de Sistemas ERP; de manera que permita incrementar la calidad de la información y reducir los tiempos utilizados en solucionar los problemas de toma de decisiones de estos proyectos.

Y ante la necesidad de las empresas dedicadas a implementar sistemas ERP de tomar decisiones eficientes ante los problemas o situaciones que se presenten en los proyectos, se hace pertinente que estas se encuentren apoyada por herramientas que le permitan dar solidez a la decisión tomada por parte del personal encargado de dirigir el proyecto, adaptándose a las características particulares de cada caso y muchas veces partiendo de información imprecisa e insuficiente, por lo que usar procedimientos manuales no es suficiente para obtener resultados significativos.

Por lo tanto, se considerará para el desarrollo del proyecto los siguientes puntos:

- **Captura de Conocimiento:** Para la Captura de Conocimiento se recopilará información sobre el conocimiento para la toma de decisiones en los proyectos de implementación del ERP Oracle E-Business Suite, para lo cual este proceso recogerá la experiencia acumulada por cada uno de los profesionales en soluciones eficientes ante los problemas o situaciones que se presenten en los proyectos de implementación este ERP en los módulos financieros. En los tres casos analizados se plantea recuperar el conocimiento en base a la experiencia o conocimiento de los expertos en el tema, para este paso se realizará de la misma forma pero adicionalmente se utilizara una metodología de Modelo de captura de conocimiento incluida en el punto 2.9 del Capítulo II.

- **Arquitectura:** Presentará una Arquitectura que refleja el ciclo de vida del CBR recuperar, reutilizar, revisar y retener, para lo cual se apoya en el uso de la herramienta jColibri que contiene estos procesos en su estructura y además servirá para validar la solución para el caso de estudio de este trabajo. Esta solución esta basada en el análisis de los casos I y III los cuales toman el apoyo en la Herramienta jColibri para poder abarcar todo el ciclo del CBR y poder plasmarlo en su Arquitectura.

- **Base de Conocimientos:** Se creará una base de conocimientos estructurada y se tratara de vincular cada caso de la base con las clases correspondientes de la ontología, con el fin de realizar una búsqueda eficaz y obtener los mejores resultados. Esta Solución es adaptada de los casos I y III analizados ya que cada una plantea una base de conocimientos estructurada que sea capaz de relacionarse con la Ontología planteada, del Caso II se tomará en cuenta crear esta base de conocimientos en relación a la experiencia o conocimiento del tema, lo cual también esta incluido en la Ontología.

- **Gestión del Conocimiento:** Para la Gestión del Conocimiento se definirá una ontología para el caso, la ontología que se creará para esta aplicación considerará empezar estructurando y categorizando los casos, de manera que puedan crearse vistas de acuerdo con indicadores definidos. Los casos recopilados serán estructurados de

acuerdo con las condiciones necesarias para que se configure y las situaciones que deben evidenciarse para que se cumpla cada condición, tratando de estructurarla jerárquicamente. La información que se recopile para la ontología a definir será aplicable a la toma de decisiones en los proyectos de implementación de sistemas ERP basándose en las experiencias anteriores. Los tres casos analizados utilizan una metodología para Gestionar el Conocimiento pero principalmente se tomará la solución aplicada por los casos I y III, los cuales usan una Ontología para este punto.

- **Herramienta Utilizada:** Como herramienta de apoyo se utiliza el jColibri para crear la solución para este caso presentado. Esta Solución es tomada del Análisis de los Casos I y III, ya que se considera que la herramienta más adecuada por su eficiencia y fácil aplicación para estos tipos de modelos que contienen RBC es el jColibri.

Finalmente este prototipo podrá usar parte de la información agrupados en una base de casos, de manera que cuando necesitan resolver algún problema se busca el caso más similar y se presentan al usuario adaptándolo o retornándolo como referencia usando los siguientes subprocesos que serán provistos para la herramienta jColibri:

- Recuperar los casos más similares
- Reusar la información y el conocimiento
- Revisar la solución propuesta
- Retener las partes de esta experiencia que podrán ser útiles en el futuro.

A continuación en los siguientes puntos se describe detalladamente cada punto del Planeamiento de la Solución del Problema descrita en este punto.

4.3 Captura de Conocimiento

Para la Captura de Conocimiento se recopilará información sobre el conocimiento para la toma de decisiones en los proyectos de implementación del ERP Oracle E-Business Suite, para lo cual este proceso recogerá la experiencia acumulada por cada uno de los profesionales en soluciones eficientes ante los problemas o situaciones que se presenten en los proyectos de implementación este ERP en los módulos financieros.

Para lo cual se tomará las siguientes Fuentes de Datos:

- **Fuentes de Datos Primarias:** El análisis se realizará en la empresa HSP S.A.C. a través de visitas para la observación de los procesos, además de realizar entrevistas al personal.
- **Fuentes de Datos Secundarios:** Para obtener esta información se considera como primera opción la consulta de la base de datos interna de la empresa para obtener datos cronológicos partiendo desde la formación hasta llegar a la estructura organizacional actual. Adicionalmente se adquirió información en libros y artículos especializados en el tema, así como páginas Web para obtener información actualizada sobre el giro del negocio.

La captura del conocimiento se inicia con la documentación de los procesos, para lo cual se ha considerado realizar una captura del conocimiento de tácito a explícito como lo menciona Sáez Vacas [18], denominado codificación. Para llevar a cabo ello se hará uso de medios electrónicos para el respaldo de la información y que ésta esté disponible en todo momento. La captura de los conocimientos tácitos debe empezar por definir un mapa de conocimiento, que permita a la organización ubicar a las personas adecuadas para la consulta en la solución de problema específico. Una vez realizado este mapa se procederá a seleccionar expertos en aquellas actividades que formen parte de un mismo proceso e iniciar un proceso de codificación, para ello se puede emplear: mapeo, tablas de decisión, árbol de decisiones, descripción por medio de reglas o recetas, modelos, Razonaminento Basado en Casos (CBR) y marcos (frames); el método a escoger se seleccionará en base a las necesidades de información de la organización.

Para el método CBR se recomienda la implementación de este en el área operativa o principal de la empresa HSP S.A.C, el cual es la implementación de sistemas ERP Oracle E-Business Suite. Este método se basa en que las soluciones a los nuevos problemas presentados en este ámbito de trabajo surgen de hechos pasados, y que al representar nuevamente el mismo problema o un problema similar en los proyectos de implementación de sistemas ERP, se pueda aplicar la solución predeterminada en base a las experiencias pasadas. Por lo tanto puede ser una excelente herramienta para los trabajadores en la solución de problemas frecuentes y no tener que esperar al experto.

Una vez codificado este conocimiento deberá seleccionarse y ser condensado en manuales de consulta rápida, además de garantizar que el personal pueda tener acceso.

Para la Captura del Conocimiento y así como su representación se tendrá en cuenta un Modelo para Representación de Memoria Organizacional en Bases a Casos.

4.3.1 Modelo para la Representación de una Memoria Organizacional en Base a Casos

El conocimiento que posee la empresa HSP S.A.C. en los problemas de implementación de sistemas ERP es muy grande y muchas veces no se encuentra representado en alguna estructura que permita consultarlo en cualquier momento, el tipo de conocimiento que se representara para este caso de estudio, trata sobre los conocimientos necesarios para la resolución de problemas en los módulos financieros del ERP Oracle E-Business Suite, así como también los problemas referidos a la Gestión del Proyecto de Implementación de este sistema en una Empresa X.

Una de las maneras de lograrlo, es guardar esta experiencia por medio de casos.

La razón de enfocarse a casos, se debe a que a través de ellos puede transferirse gran parte del conocimiento que actualmente no se tiene documentado y que contiene experiencias, habilidades y competencias del trabajo realizado por el personal de la empresa que le da a la organización ventajas competitivas.

Para la representación formal de los Casos se empleará el Modelo para Representación de una Memoria Organizacional en Base a Casos, descrito en el punto 2.9., del Capítulo II.

Por lo tanto, a continuación se describen cómo se aplicarán los siguientes puntos para nuestro caso de estudio.

4.3.1.1 Casos

Un caso es una pieza contextualizada de conocimiento que representa las experiencias obtenidas en proyectos de implementación del ERP Oracle E-Business Suite en proyectos pasados donde se resolvió algún problema presentado.

Típicamente un caso para estos problemas comprende los siguientes puntos:

- Tipo de Caso, que Identifica un caso.
- Modulo, a que modulo del ERP pertenece.
- Temática, subclasificación del caso referente al Modulo Descrito.
- Nivel de Riesgo, con tres niveles posibles (Alto, Medio, Bajo).
- Método de Evaluación, con cuatro enfoques definidos (Directivo, Consultivo, Democrático y Consenso).
- Acción Tomada, que indica cual fue el uso del caso recuperado.
- El problema, que describe el estado del mundo cuando ocurrió el caso
- La solución, que establece la solución derivada de ese problema
- El resultado, que describe el estado del mundo después de ocurrido el caso

El modelo que se utilizará para la representación formal de un caso, tendrá la siguiente estructura adaptada al caso de estudio (Ver Figura 4.2.).

Script

Nombre:
Tipo de Caso:
Modulo:
Temática:
Nivel de Riesgo:
Método de Evaluación:
Acción Tomada:

Escena 1: Problema

Escena 2: Solución

Escena 3: Resultado

Fig. 4.2. Modelo de Representación en Script de un Caso Tipo.

A continuación se muestra un ejemplo de un caso representado bajo este esquema:

Script Proveedor no Disponible en Ingreso de Facturas

Nombre:	Proveedor no Disponible en Ingreso de Facturas
Tipo de Caso:	Funcional
Modulo:	Cuentas por Pagar
Temática:	Facturas por Pagar
Nivel de Riesgo:	Alto
Método de Evaluación:	(NO Aplica)
Acción Tomada:	Aceptada

Escena 1: Problema
Al Registrar o Ingresar una Factura, un determinado proveedor no esta en lista de valores a pesar de que existe en el maestro de proveedores del sistema.

Escena 2: Solución
Se debe registrar la sucursal del proveedor, a pesar de que un proveedor exista en el maestro, debe tener registrada una sucursal en la organización actual para que este se pueda mostrar en la lista de valores de proveedores desde la pantalla de Ingreso de facturas.

Escena 3: Resultado
El proveedor en cuestión ya se encuentra disponible como dato de entrada en la lista de valores para ingresarlo en la factura.

Fig. 4.3. Ejemplo de Representación en Script de un Caso Tipo.

4.3.1.2 Modelo

En la Figura 4.4., se muestra el modelo propuesto para la representación del conocimiento inmerso en casos de los problemas presentados en proyectos de implementación del ERP Oracle E-Business Suite. Este modelo está formado por tres partes principales: Casos, Representación formal de casos y su Representación por medio de herramientas computacionales.

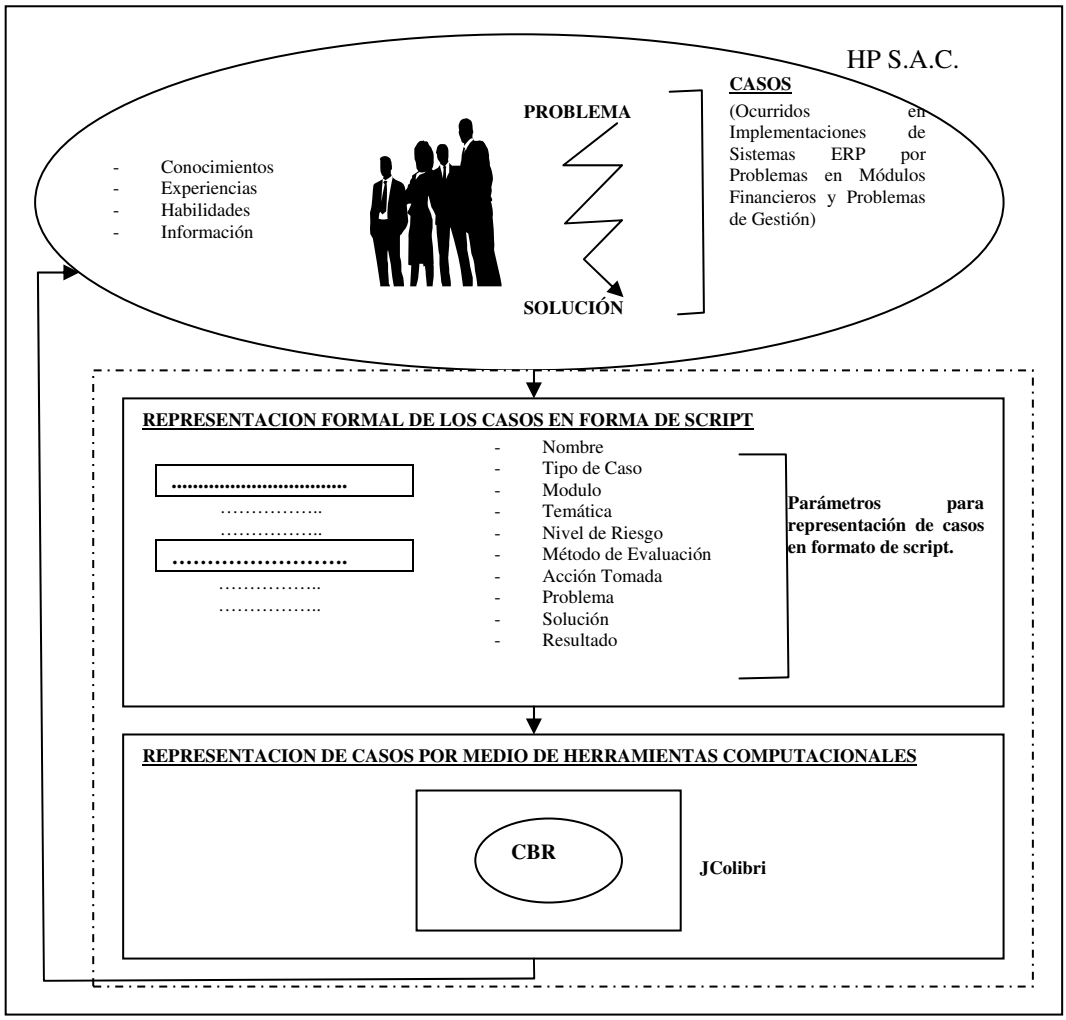


Fig. 4.4. Modelo de Representación de una Memoria Organizacional.

La primera parte del modelo, casos, especifica el tipo de conocimiento que interesa representar. Este conocimiento está en forma de caso y tiene todas aquellas experiencias pasadas que han permitido resolver problemas en problemas en implementaciones realizadas del ERP Oracle E-Business Suite. Estos casos deberán pasar a formar parte de una memoria organizacional compuesta por la representación formal de casos y la representación por medio de herramientas computacionales.

La finalidad de que los casos se documenten de manera formal, es para facilitar su entendimiento y manipulación. Cuando se haya realizado la representación formal de los casos de acuerdo a la estructura propuesta en la Figura 4.2., se deberá pasar a la tercera parte del modelo en la que se buscará la representación por medio de herramientas computacionales por lo cual su representación para nuestro caso será en la Base de Casos del CBR utilizando el jColibri para este propósito.

Con el uso del jColibri se permite la captura y facilita el acceso al conocimiento inmerso en los casos por todos los consultores y Jefes de Proyectos de la empresa HP S.A.C lo que contribuirá en el mejoramiento de la competitividad de la organización.

4.4 Arquitectura del Sistema

A continuación se presenta la arquitectura del Sistema y posteriormente se procederá a explicar cada uno de sus componentes.

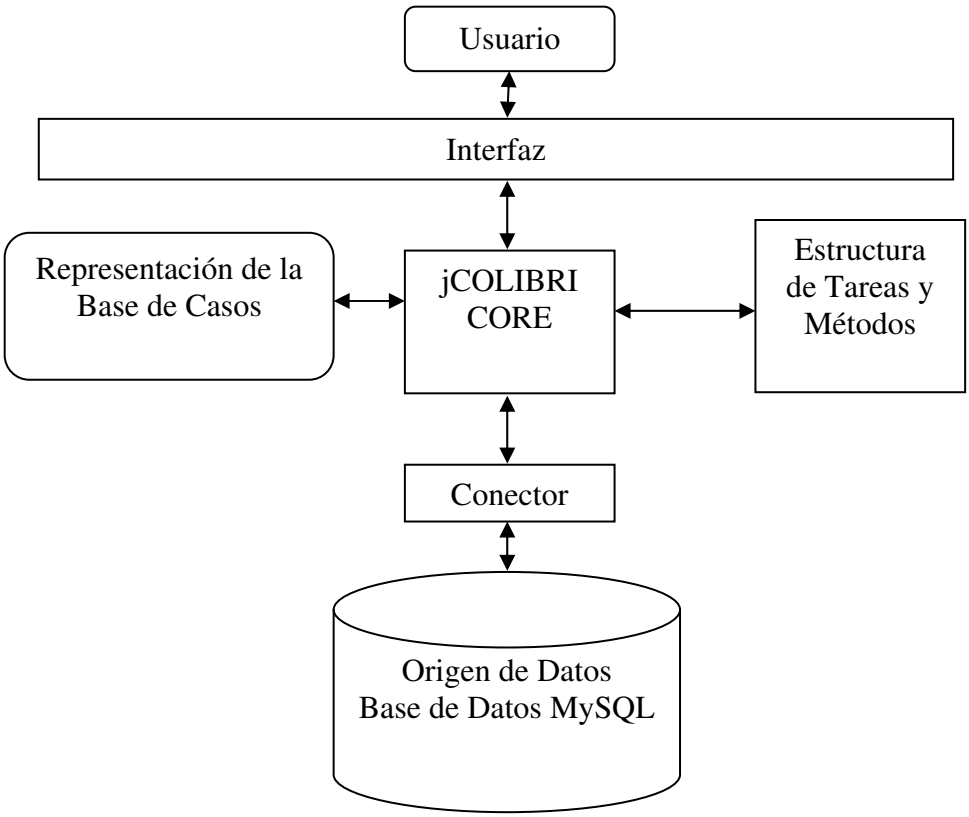


Fig. 4.5. Arquitectura del Sistema.

Esta arquitectura está basada principalmente en el diseño de jColibri, la herramienta utilizada para la implementación de esta aplicación, siguiendo la metodología propuesta por ese Armazón. A continuación se explican los principales componentes de esta arquitectura así como sus funciones principales.

- **Origen de Datos:** Corresponde a la capa de persistencia, este es el medio donde se encuentra almacenada la Base de Casos, con sus respectivos atributos, valores, descripciones, soluciones y explicaciones. Para el caso de nuestra aplicación el Origen de Datos es una base de datos MySQL, en la cual se encuentran las tablas que modelan el esquema de la Base de Casos.

- **Conector:** Los conectores son clases del armazón que ofrecen una única interfaz para el acceso a la capa de persistencia, sea cual sea el origen de datos a utilizar. Para el caso concreto de la aplicación, se está utilizando una clase que implementa la conexión con una Base de Datos, cuya configuración se encuentra almacenada en un archivo XML, el cual se crea al momento de configurar la aplicación mediante la herramienta jColibri. En este archivo se especifica la dirección del sistema gestor de la base de datos, las credenciales de autentificación y el mapeo entre atributos del caso y columnas de la tabla que los almacena.

- **Representación de la Base de Casos:** Este componente hace referencia a la representación de los casos en memoria y su organización, jColibri incluye diversas

implementaciones de la base de casos donde los casos se indexan y estructuran mediante diferentes tipos de datos: listas, árboles, etc. La estructura que se utilizará será la de árbol, donde los nodos internos son atributos compuestos y las hojas atributos simples. Adaptando este esquema a la representación de los casos, los elementos a utilizar se denominan individuos (Individual) y representan un atributo del caso. A su vez, cada uno de estos individuos puede estar compuesto de otros individuos. La composición entre individuos se realiza a través de objetos denominados IndividualRelation que representan las relaciones entre individuos.

- **jColibri CORE:** Este componente engloba principalmente a los métodos encargados de manipular los casos e implementar las distintas etapas del ciclo CBR. Aquí se define clases como Caso, Base de Casos, Consulta, Similitud, etc.

- **Estructura de tareas y métodos:** Se define la estructura de las tareas y los métodos para resolver estas tareas, dicha estructura se encuentra almacenada en un archivo XML, el cual se genera al momento de configurar este componente con la herramienta jColibri. Las tareas se estructuran jerárquicamente de forma que un conjunto de subtareas descomponga la funcionalidad de la tarea superior. Las principales tareas de esta organización son Retrieve, Reuse, Revise y Retain (Recuperación, Reutilización, Revisión y Retención) siguiendo el famoso modelo presentado por Aamodt y Plaza [9]. Dichas tareas son descompuestas en secuencias de otras tareas recursivamente hasta que una tarea no puede ser descompuesta en acciones más sencillas y es resuelta por uno de los métodos incluidos en el armazón. La información a representar en el caso de las tares consiste simplemente en el nombre de la misma y su descripción. La definición de los métodos es mucho más extensa y en ella se incluye la siguiente información:

    - Nombre del método.
    - Descripción.
    - Precondición. Representa las condiciones necesarias para aplicar el método.
    - Tipo de método: descomposición o resolución.
    - Parámetros. Sirven para adaptar los métodos al dominio concreto de la aplicación.
    - Competencias. Indican las tareas que el método puede resolver.
    - Subtareas. Especifican las tareas que un método de descomposición genera.
    - Postcondición. Representa el estado de la aplicación tras aplicar el método.

- **Interfaz:** En esta componente se encuentran las interfaces graficas que se le presentan al usuario para que este realice consultas en la aplicación y obtenga los resultados o recomendaciones del sistema.

4.5 Base de Conocimiento

La aplicación del sistema se inicia con la recopilación de información de casos de las prácticas recuperadas en la base de conocimientos de acuerdo a las fuentes (empresa HSP S.A.C.) tomadas según la descripción de los pasos para la recuperación del conocimiento, con el fin de poblar la herramienta CBR JColibri.

Los casos están estructurados con las siguientes propiedades:

- **Tipo de Caso:** Identificación de un caso, puede ser de dos tipos diferentes: Funcionales (Métodos, Procedimientos, Configuraciones, etc.) o de Gestión (Planificación de Tiempos, Requerimientos, etc.).

- **Módulo:** Describe el Módulo del ERP al que pertenece (Administración, Compras, Contabilidad, Cuentas por Pagar, Cuentas por Cobrar, Inventarios y una Categoría especial de Gestión para especificar los Tipos de Caso de Gestión).

- **Temática:** Subclasificación del caso referente al Módulo Descrito, indica la instancia de la Clase a la que pertenece (Ejemplo: Facturas por Pagar, Órdenes de Compra, Habilitación de Proveedores, etc.)

- **Nivel de Riesgo:** Esta propiedad es sólo para los tipos de caso de Gestión y puede tener solo un nivel de riesgo de los tres niveles posibles (Alto, Medio, Bajo).

- **Método de Evaluación:** Esta propiedad es sólo para los tipos de Caso de Gestión y puede ser de cuatro enfoques definidos (Directivo, Consultivo, Democrático y Consenso).

- **Acción Tomada:** Indica cual fue el uso del caso recuperado, para lo cual tiene tres valores Aceptada, Rechazada o Faltan Detalles.

- **Problema:** Da las ideas, características o descripciones principales del Problema que se tiene en la base de casos.

- **Solución:** Da las ideas principales para tratar el caso en relación con los valores descritos anteriormente según el Tipo de Caso al que pertenece (Gestión o Funcional).

- **Resultado:** Da las ideas principales y descripciones del resultado obtenido en la aplicación de este caso de la base de conocimiento, en la toma de decisiones de acuerdo a la situación presentada.

Una visión de un conjunto de la base de conocimientos se ilustra en la Tabla 4.1.

TIPO DE CASO	MODULO	TEMATICA	NIVEL DE RIESGO	METODO DE EVALUACION	ACCION TOMADA	PROBLEMA	SOLUCION	RESULTADO
FUNCIONAL	CUENTAS POR PAGAR	FACTURAS POR PAGAR	ALTO			AL INGRESAR UNA FACTURA, UN DETERMINADO PROVEDOR NO ESTA EN LA LISTA DE VALORES A PESAR DE QUE ESTE EXISTE EN EL MAESTRO DE PROVEEDORES	SE DEBE REGISTRAR LA SUCURSAL DEL PROVEEDOR, A PESAR DE QUE UN PROVEEDOR EXISTA EN EL MAESTRO, DEBE TENER REGISTRADA UNA SUCURSAL EN LA ORGANIZACIÓN ACTUAL PARA QUE ESTE SE PUEDA MOSTRAR EN LA LISTA DE VALORES DE PROVEEDORES DESDE LA PANTALLA DE INGRESO DE FACTURAS	EL PROVEEDOR EN CUESTION YA SE ENCUENTRA DISPONIBLE COMO ENTRADA EN LA LISTA DE VALORES PARA INGRESARLO EN LA FACTURA
FUNCIONAL	CUENTAS POR COBRAR	FACTURAS POR COBRAR	ALTO			AL INGRESAR LAS LINEAS DE UNA FACTURA Y GUARDAR LOS DATOS, LAS DISTRIBUCIONES DE LA FACTURA PRESENTAN LINEAS CONTABLES CON SEGMENTOS INCOMPLETOS.	REVISAR LA PANTALLA DE CONFIGURACIÓN AUTOMATICA, BUSCAR EL TIPO DE LINEA DE DISTRIBUCION QUE ESTA PRESENTANDO EL PROBLEMA, Y AHÍ SE OBSERVA CUAL ES EL ORIGEN DE LOS VALORES DE CADA SEGMENTO CONTABLE, SI ALGUNOS SEGMENTOS ESTAN INCOMPLETOS SE DEBE A QUE SE DEBE SETEAR EL VALOR EN EL ORIGEN DE DICHO SEGMENTO	AL GUARDAR LA FACTURA SE COMPLETARAN AUTOMATICAMENTE LOS SEGMENTOS DE LAS CUENTAS A COBRAR, INGRESOS E IMPUESTOS.
GESTION	GESTION	GESTION DE TIEMPOS	MEDIO	DIRECTORIAL	ACEPTADA	SE TIENEN NUEVAS DEFINICIONES O REQUERIMIENTOS NO CONTEMPLADOS EN EL ALCANCE DE PROYECTO O CONTRATO INICIAL	ESTOS NUEVOS REQUERIMIENTOS QUE REQUIEREN CAMBIOS EN LA PLANIFIFICACION DEL CRONOGRAMA DEL PROYECTO SERAN TOMADOS COMO ADICIONALES A COTIZAR PARA EL CLIENTE LUEGO DEL PROYECTO, YA QUE NO ESTABAN CONSIDERADOS DENTRO DEL CONTRATO INICIAL POR LO CUAL NO SE CONSIDERO UNA PLANIFICACION PARA SU DESARROLLO EN ESTOS PERIODOS.	LA DECISION TOMADA FAVORECE EL FLUJO DEL CRNOGRAMA EN SU ESTRUCTURA PLANIFICADA YA QUE NO SE REALIZA NINGUNA MODIFICACION A EL Y TODO SIGUE DE ACUERDO A LO PLANIFICADO EN GESTION DE TIEMPO.

Tabla 4.1. Ejemplos de Base de Conocimientos.

En la tabla 4.1., se muestran ejemplos de casos almacenados bajo el esquema propuesto, la descripción principal del problema será almacenado en el atributo de tipo texto “Problema”, a su vez, la información es complementada con los demás atributos que presentan valores dependiendo del tipo de caso que se está consultando. Esta representación es la que simula los casos almacenados en la base de datos.

4.6 Gestión del Conocimiento (Ontología)

Para las aplicaciones CBR, la definición clara y eficiente de una Ontología es una tarea crítica en el diseño de este tipo de sistemas. La ontología creada para esta aplicación contiene una clasificación y agrupación de los distintos problemas que se presentan en la implementación

de un Sistema ERP. Esta clasificación está basada, primero en el tipo de problema que se va a resolver (Funcional o de Gestión), y luego, dependiendo del tipo de problema, se definen una serie de atributos o temáticas para estos.

A continuación se describirán las principales clases de la ontología desarrollada para facilitar el entendimiento y expresividad del dominio del conocimiento.

La ontología, definida mediante una estructura de árbol presenta dos clases principales, las cuales representan 2 tipos de problemas que se presentan cotidianamente en un proyecto de implementación de ERP:

- **Funcionales:** Esta clase agrupa a los problemas que se presentan con la funcionalidad del ERP, tanto al momento de su configuración como en el momento de su puesta en marcha y por ende la utilización del sistema por parte de los usuarios finales.
    - Dentro de esta clasificación de problemas, se han creado clases para agrupar los problemas de acuerdo al módulo del cual provienen; como se definió en los alcances del presente trabajo, los módulos que se abarcarán serán los financieros, por lo tanto dentro de cada módulo se ha subdividido los problemas de acuerdo al componente del módulo sobre el cual recae dicho problema. De modo que se tienen las siguientes clases representativas de los módulos a tratar:
        - Administración
        - Contabilidad
        - Cuentas por Cobrar
        - Cuentas por Pagar
        - Compras
        - Inventarios
- **Gestión:** Esta clase engloba a los problemas presentados en la dirección del proyecto, tomándolo de una perspectiva a más alto nivel, se tratan de problemas que tienen que ver con la gestión propia del proyecto y cuya responsabilidad recae en las personas encargadas de dirigir dichos proyectos.
    - Para este tipo de problemas, se han definido una serie de atributos que permitan una mejor descripción del problema. En esta clase no se han separado los problemas por temáticas debido a que los problemas de gestión no pertenecen a un modulo del Sistema ERP determinado, sino que son problemas a nivel de gerencia del proyecto.

Para la elaboración de esta Ontología se ha utilizado la herramienta Protege y a continuación se muestra la pantalla donde se define la ontología:

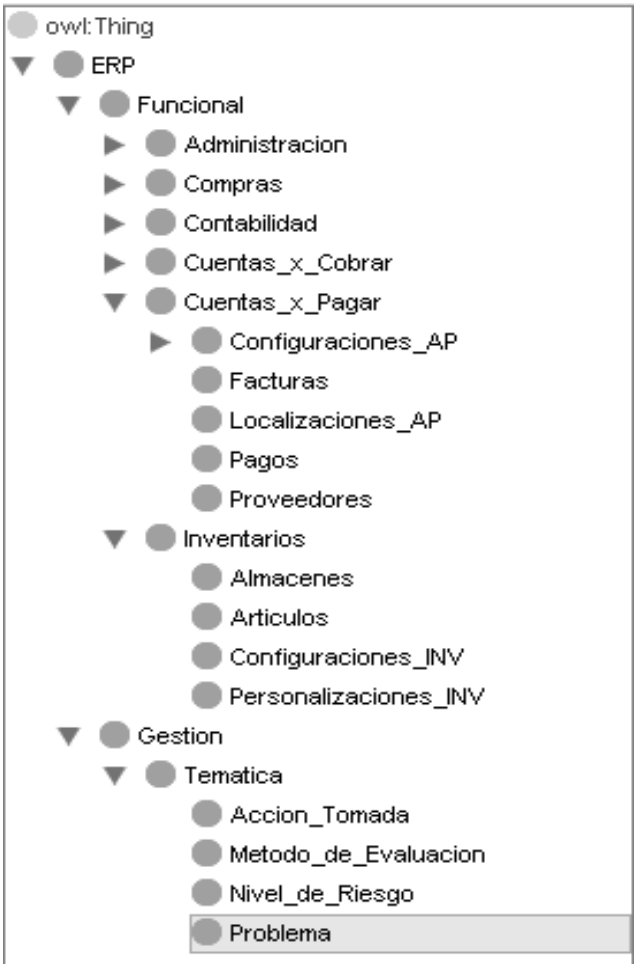


Fig. 4.6. Ontología de Problemas de Implementación de Sistemas ERP.

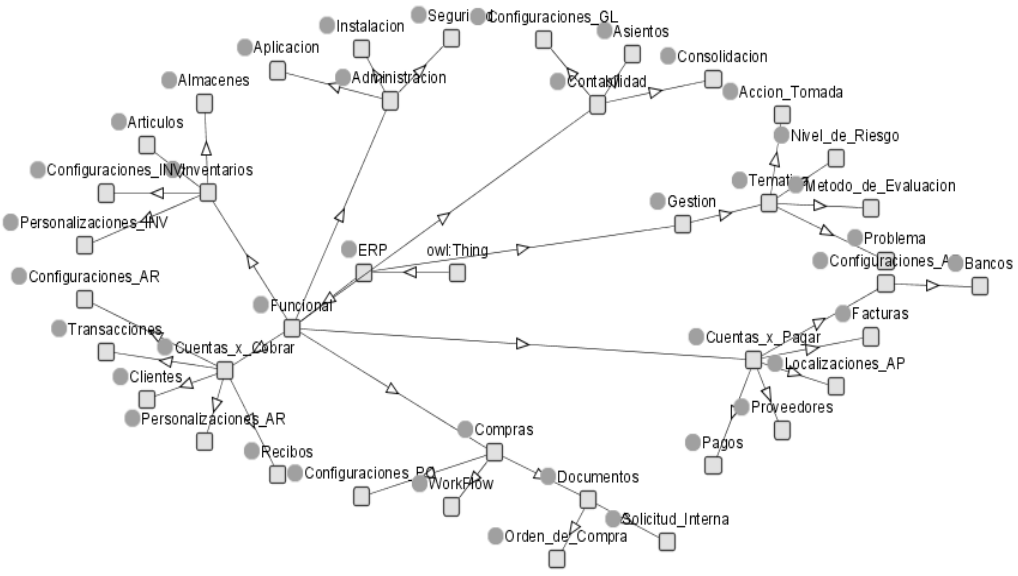


Fig. 4.7. Diagrama Jambalaya de la Ontología.

4.7 Diseño mediante jColibiri y su Procesamiento

En esta sección se describirá en forma genérica las configuraciones realizadas en la herramienta jCOLIBRI, el procesamiento que realiza la aplicación y como es que utiliza los diferentes componentes descritos en la arquitectura para obtener los resultados esperados.

Las configuraciones de diseño usando jCOLIBRI son las que se presentan a continuación:

4.7.1 Creación de nueva Aplicación

En esta pantalla se seleccionan los componentes y extensiones que estarán disponibles para el diseño del sistema. Siempre se encuentra seleccionada la opción "Core", ya que aquí se encuentra los elementos fundamentales para el procesamiento del ciclo CBR. Adicionalmente, seleccionamos las opciones "Description Logic Extension", para la utilización de la Ontología y "Textual Extension" para el procesamiento de textos (Ver Figura 4.8.).

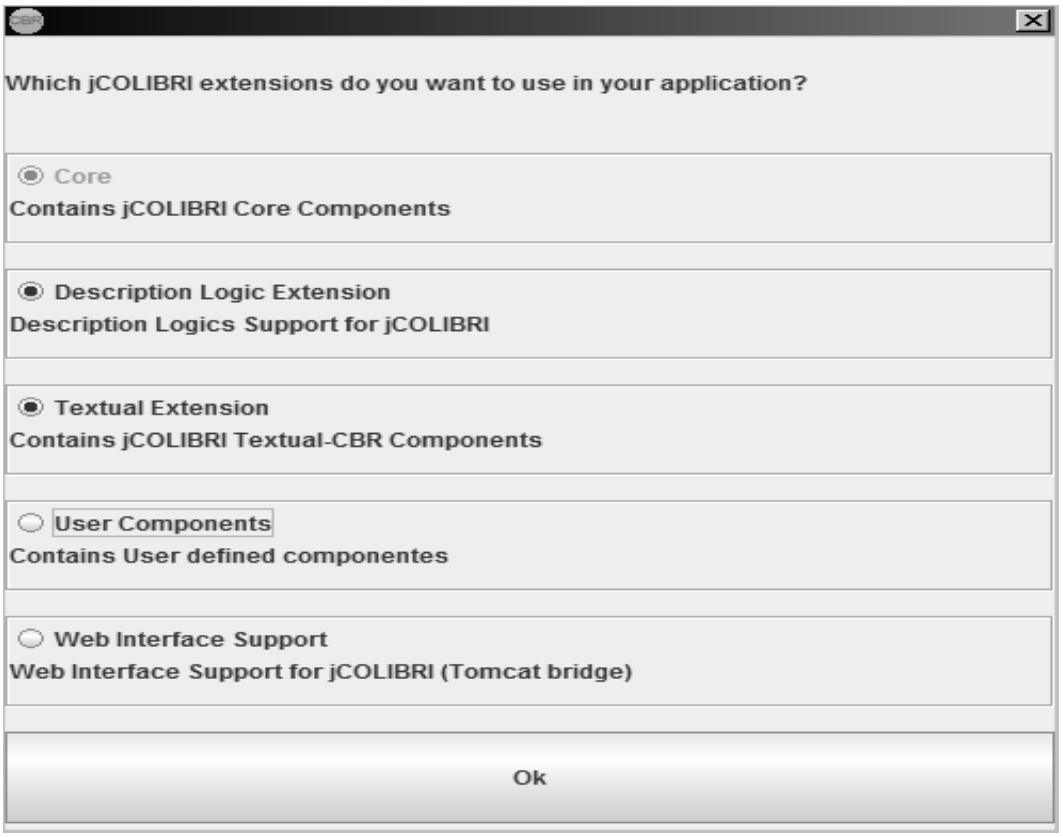


Fig. 4.8. Creación de una nueva aplicación.

4.7.2 Configuración de la Estructura de Casos

A continuación, se define la estructura de casos; tal como se describió en apartados anteriores, esta estructura será tomada por la aplicación, para definir la estructura de la consulta que se le mostrará al usuario.

Se define un atributo de tipo Textual, el cual será una breve descripción del problema presentado y que será comparado con el campo textual de la base de casos almacenada.

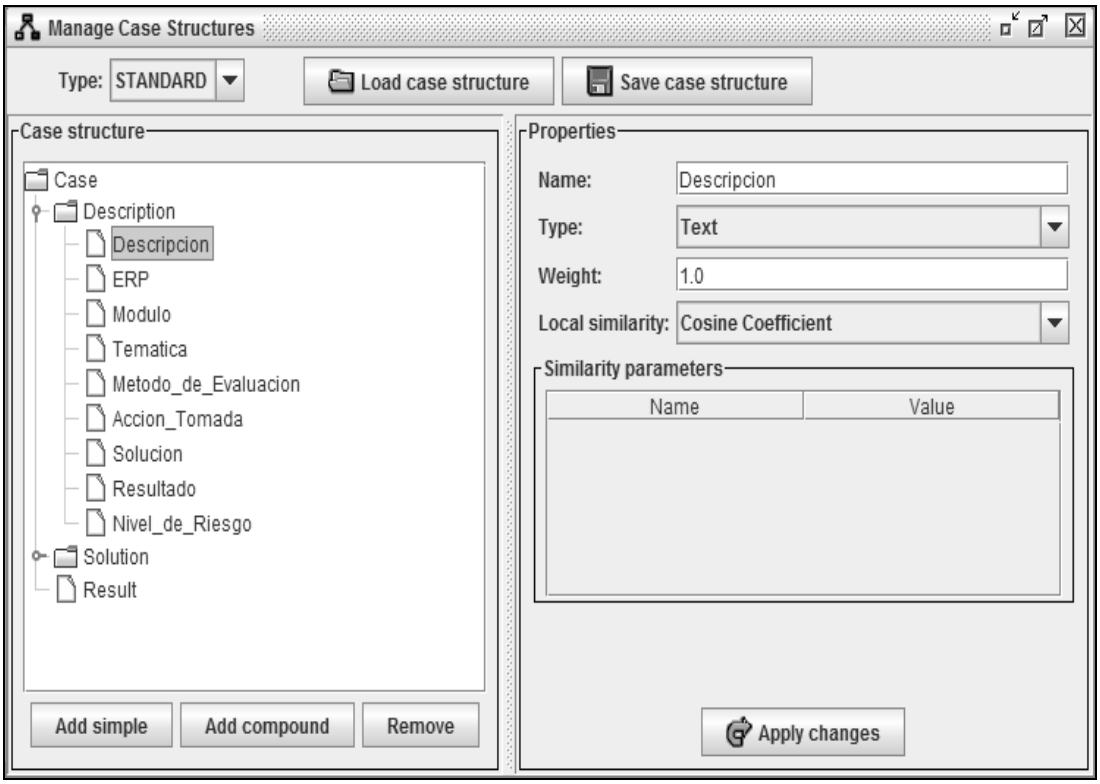


Fig. 4.9. Configuración de la Estructura de Casos.

Otros atributos de la estructura, utilizan la ontología creada para presentar una lista de valores al momento de solicitar la información al usuario, a continuación se muestra como se realiza esta configuración.

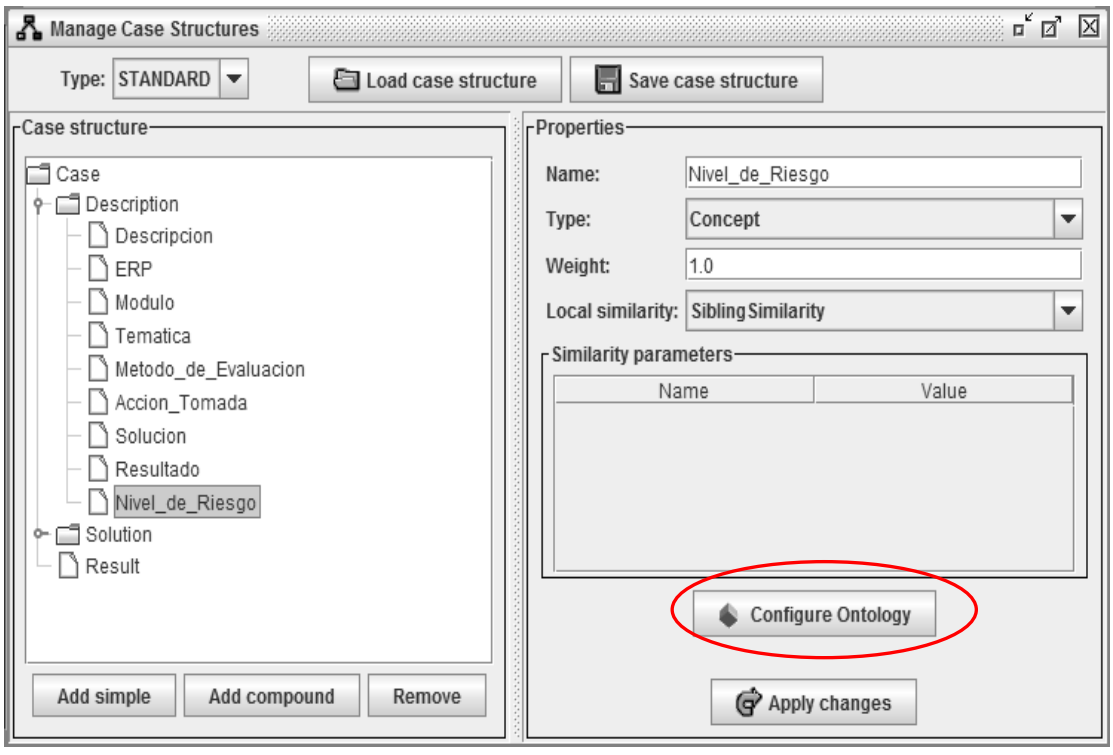


Fig. 4.10. Configuración de Atributo de tipo Concepto.

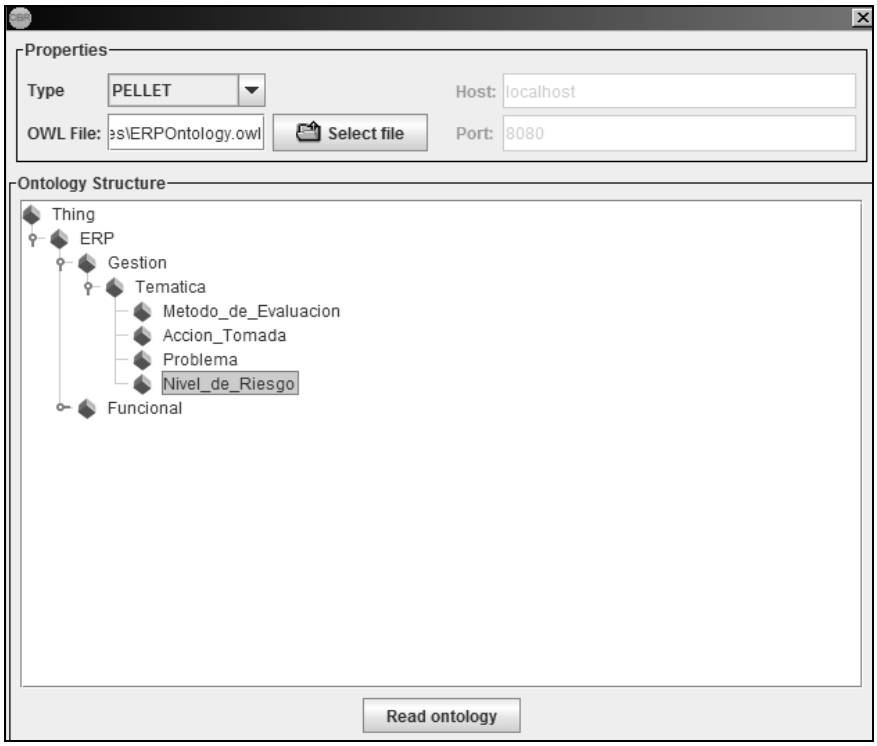


Fig. 4.11. Selección de Concepto en la ontología.

4.7.3 Configuración del Conector

Una vez definida la estructura de casos, se debe definir el conector. Para el caso de la presente aplicación se ha codificado una clase, que permite leer los casos de la base de datos MySQL y carga estos en memoria, para su posterior utilización en el procesamiento del ciclo CBR. En esta pantalla se indica la ruta del archivo XML que almacena la estructura de casos y se indica la ubicación de la clase implementada para nuestro fin (Ver Figura 4.12.).

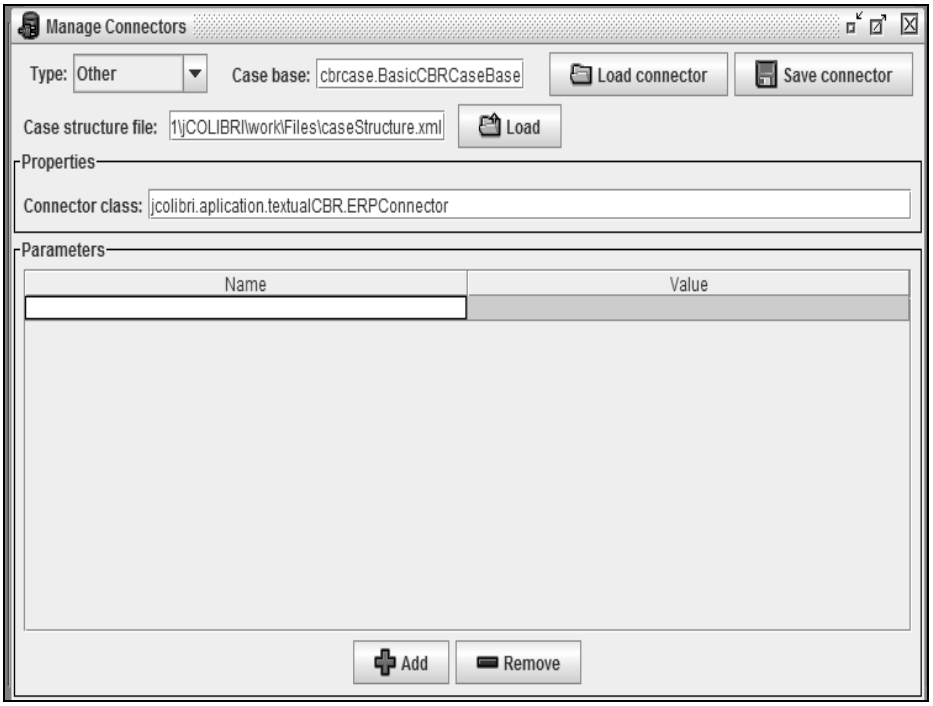


Fig. 4.12. Configuración del Conector.

4.7.4 Configuración de Tareas y Métodos

4.7.4.1 Preciclo

Se encarga de cargar los casos e inicializar los recursos necesarios. Esta parte se ejecuta una única vez antes del ciclo principal. Este proceso se hace imprescindible sobre todo en las aplicaciones de CBR Textual. Este tipo de métodos necesitan procesar el texto para extraer la información y representarla de forma estructurada, es por esto, que el procesamiento puede requerir un largo periodo de cómputo (a veces de varios minutos) por lo que sería inviable ejecutarlo cada vez que se realiza una consulta. Mediante el preciclo se permite adelantar la inicialización y carga de recursos de forma que se ejecuten los procesos más pesados antes de recibir la consulta.

A continuación, se muestra como se configura esta etapa, aquí se tiene que instanciar el método seleccionado, en esta caso se nos muestran dos posibilidades, de la cual se utilizará el método para trabajar con textos ("Simple textual pre-cycle"). Al instanciar este método, se muestran tres métodos más.

Obtain Case Task: En esta tarea se obtiene los casos de la fuente de datos. Recibe como parámetro el archivo XML que contiene los datos del conector configurado en el paso anterior.

Select working cases task: En esta tarea se seleccionan los casos con los que se va a trabajar y se almacenan en memoria, listos para ser utilizados en el procesamiento posterior.

Cases Textual Process task: Esta tarea es la que implementa el modelo de Lenz (descrito en el marco Teórico), el cual se utiliza para el procesamiento de textos.

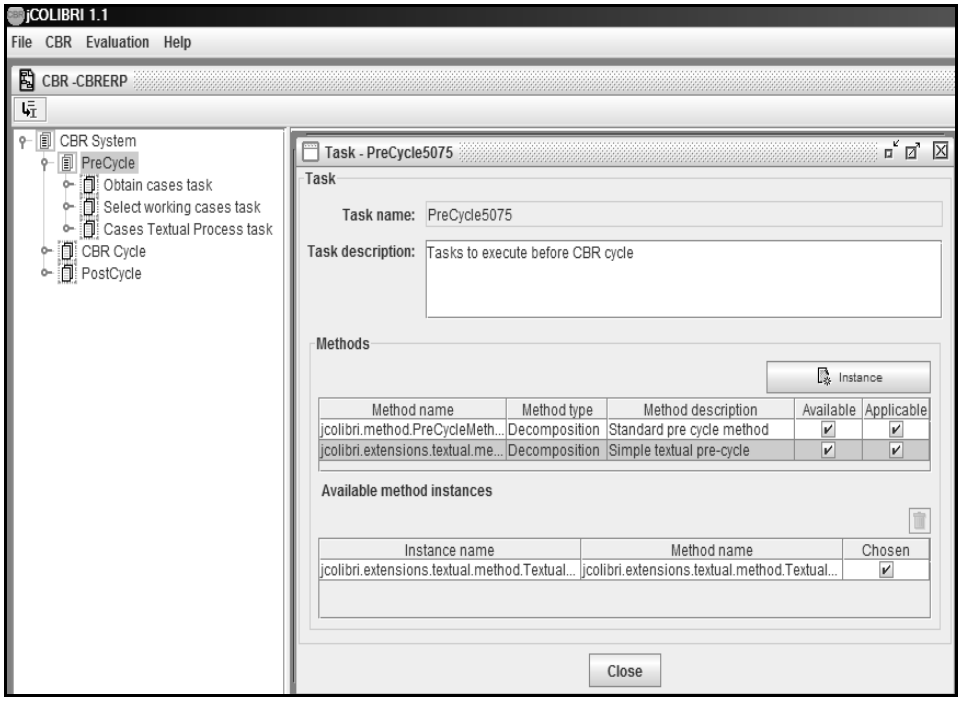


Fig. 4.13. Configuración del Pre Ciclo.

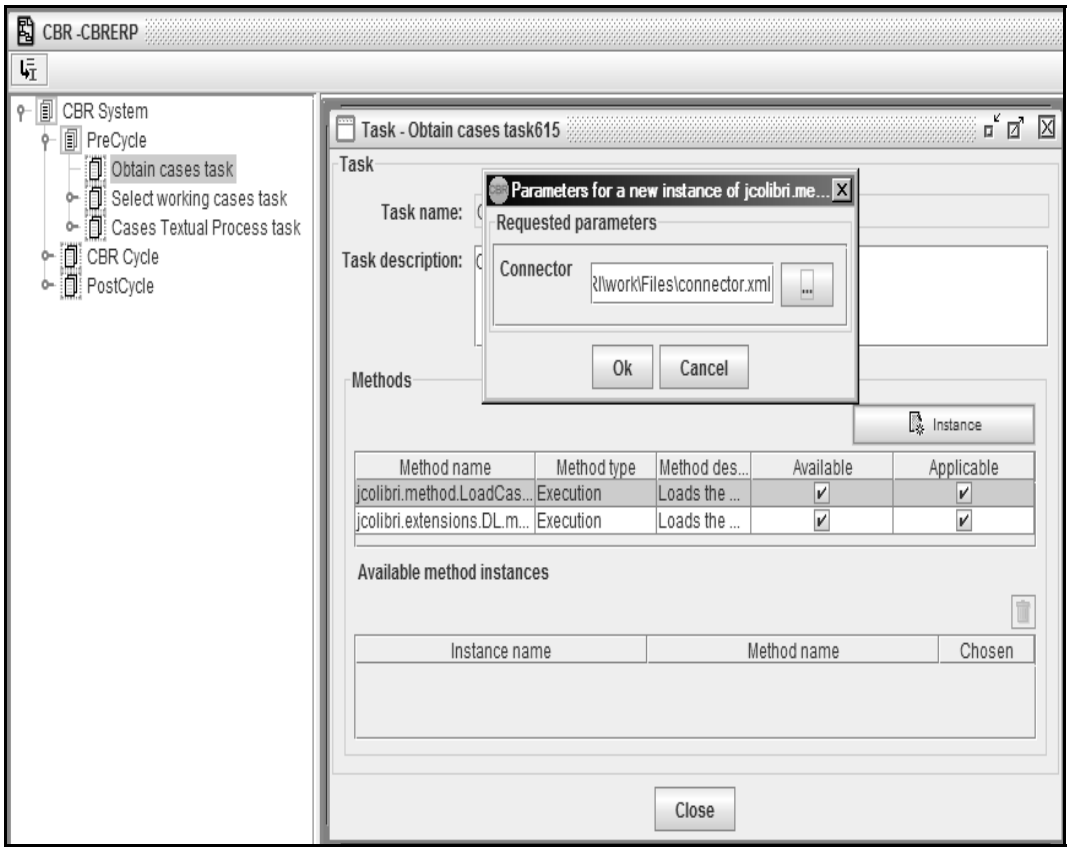


Fig. 4.14. Configuración de la tarea “Obtener Casos”.

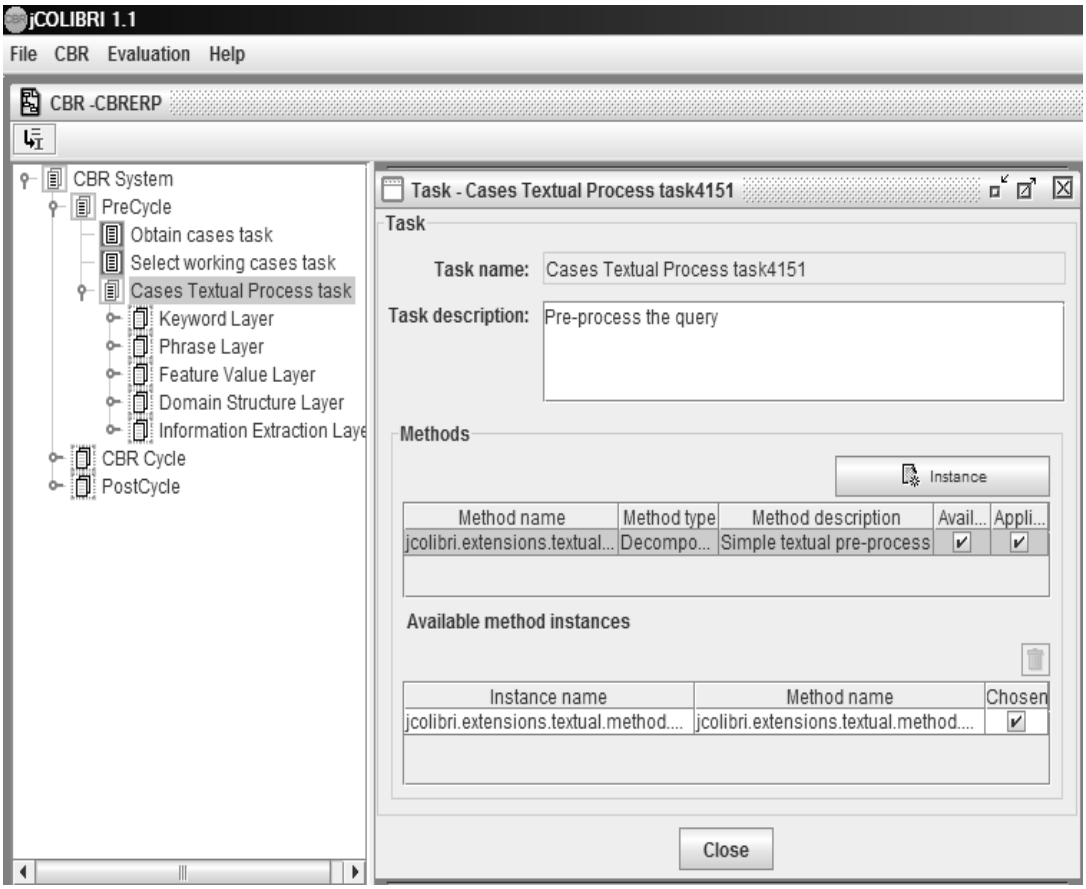


Fig. 4.15. Configuración de procesamiento de textos

4.7.4.2 Ciclo:

Se encarga de implementar la funcionalidad de cada aplicación CBR. Dentro de esta etapa se ejecuta la recuperación, adaptación y almacenamiento de los casos (aunque no todas de estas tareas son obligatorias).

A continuación, se observa que cuando se instancia esta tarea, se muestran las 4 tareas del ciclo CBR, cada una de estas tareas serán instanciadas y configuradas.

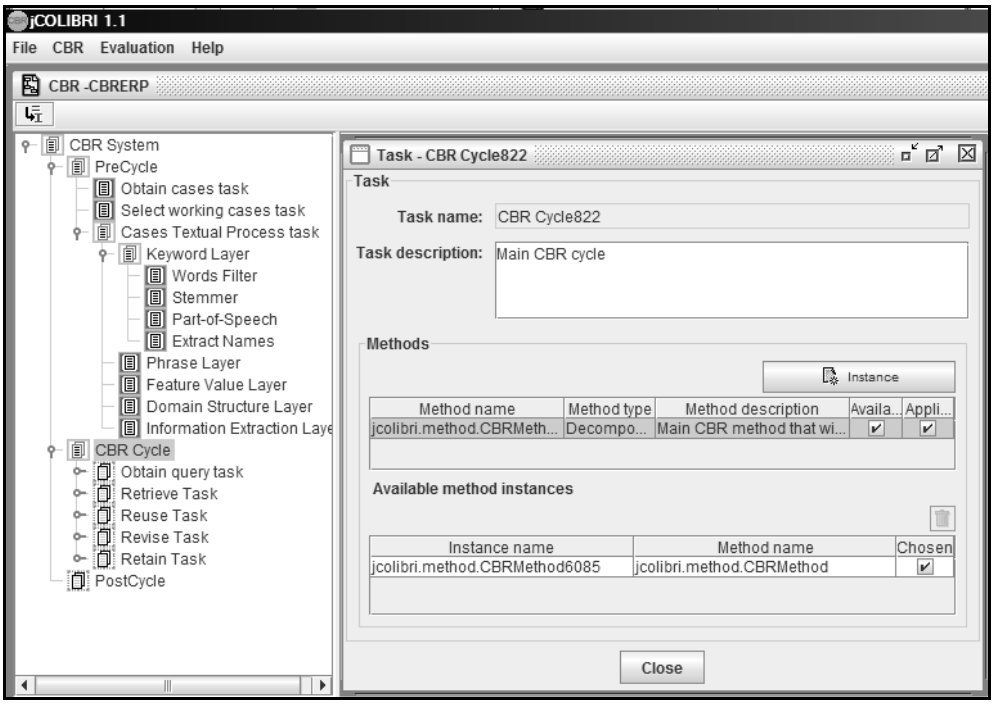


Fig. 4.16. Configuración del ciclo CBR

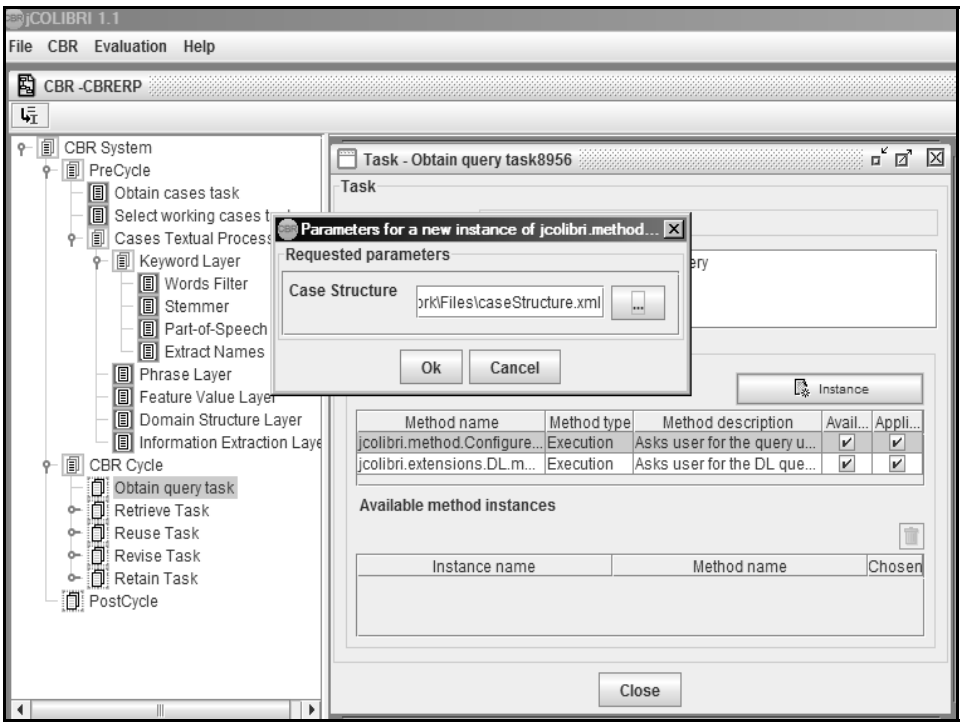


Fig. 4.17. Configuración de la tarea “Obtener query”

4.7.4.3 Postciclo:

Esta última fase se encarga de liberar los recursos de la aplicación y ejecutar los métodos de mantenimiento. Estos métodos suelen ejecutarse de forma independiente al ciclo CBR, luego de la ejecución de cierto número de ciclos (fase anterior), y se encargan de evitar la degradación en eficiencia del sistema según se incorporen nuevos casos que pudieran añadir complejidad o carga de procesamiento. Gracias al postciclo, los métodos de mantenimiento se incorporan independientemente al ciclo CBR de forma natural sin separarlos del proceso de desarrollo de todo el sistema.

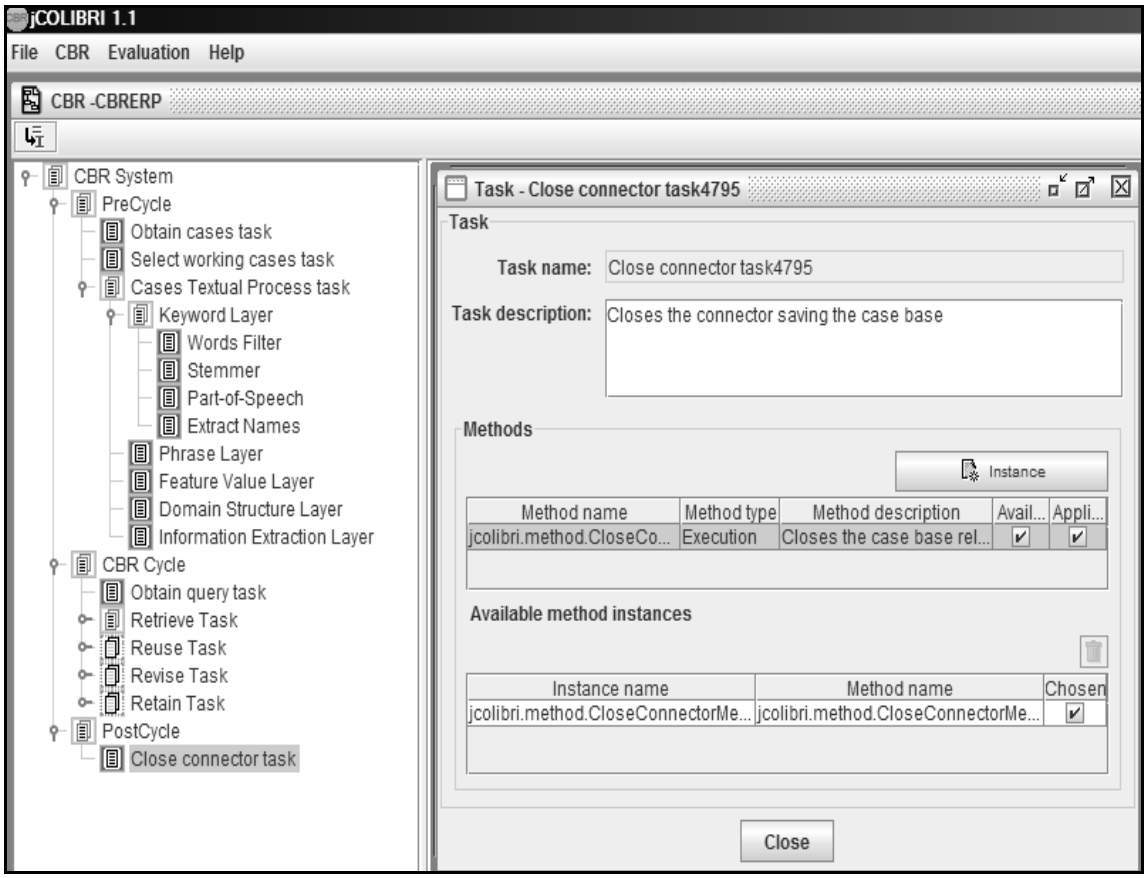


Fig. 4.18. Configuración del post ciclo

Cada una de las tres etapas anteriores se representa como una tarea normal que es resuelta por métodos de descomposición en subtareas apropiadas configuradas en el jColibri. Aunque al ejecutar la aplicación estas tres tareas principales se ejecuten de forma independiente una de la otra, esta forma de representarlas en el Framework mantiene coherencia con el mecanismo principal de diseño basado en descomposición de tareas utilizado por jCOLIBRI, lo que permite que en conjunto estas tres tareas colaboren con el logro del objetivo de la tarea principal.

4.7.5 Ciclo del CBR en el jColibri

En la figura 4.19., se muestra la representación del modelo del Ciclo del CBR descrito por Ammod y Plaza [9] en el Punto 2.4.2., contenido en el Capítulo II.

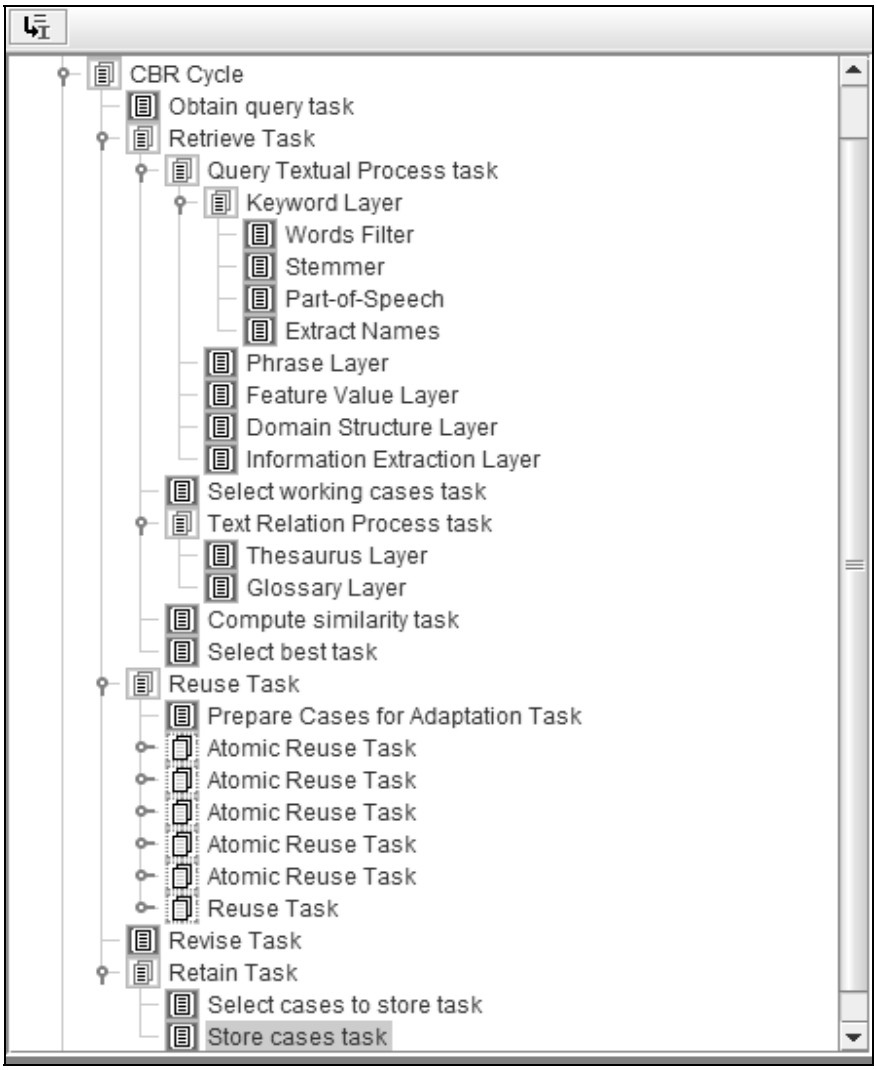


Fig. 4.19. Representación del Ciclo CBR en jColibri.

Está representación sigue el modelo de Jerarquía de Tareas, ambos conceptos fueron descritos en el capítulo 2. El modelo de Aamod y Plaza [9], se encuentra incluido en el CBR Cycle (descrita en el punto 4.7.4.2. de este capítulo), este proceso se divide principalmente en 4 tareas que representan las 4 etapas del modelo mencionado, a su vez cada tarea se descompone en más tareas que colaboran con el logro del objetivo de la tarea principal.

CAPITULO V

IMPLEMENTACIÓN PRÁCTICA

5.1 Introducción

En este capítulo se explica la manera cómo se construirá el prototipo para la herramienta de Razonamiento Basado en Casos (RBC) para toma de decisiones en proyectos de implementación de Sistemas ERP utilizando jColibri, en esta sección se definirá su arquitectura a implementar y se explican cada uno de los componentes a crear. Para un mejor seguimiento se dividió en las dos partes a implementar: análisis y desarrollo.

También se detalla las herramientas tanto de software y hardware que se necesitarán para poder implementar el sistema en una siguiente etapa, después de que se haya terminado la etapa de desarrollo y las evaluaciones o pruebas respectivas que se realizará en los trabajos futuros.

5.2 Análisis:

La fase de análisis es la etapa previa del proceso de desarrollo en la que se buscará definir con la mayor claridad posible lo que esperamos obtener, es decir, la herramienta de apoyo para toma de decisiones en proyectos de implementación de Sistemas ERP.

5.2.1 Descripción del Proyecto

La empresa en la que se llevó a cabo la implementación, se encuentra ubicada en la ciudad de Lima, está dedicada al trabajo de consultoría en Tecnología de Información, como Partner de Oracle se especializa en la implementación del ERP E-Business Suite, así como personalizaciones y localizaciones a medida del cliente. La experiencia y calificación de sus consultores permite realizar un análisis del nivel de tecnología implantado en sus clientes y recomendarle las mejores soluciones, necesarias para el éxito de su negocio con un alto valor agregado y con visión de futuro. El proceso más importante de esta empresa es la Implementación del Sistema ERP e-Business Suite.

Para el desarrollo del proyecto se seguirá la siguiente metodología, Metodología para la Implementación de una Memoria Organizacional en Base a Casos, la cual ha sido descrita en el Capítulo II, en el punto 2.9.7.

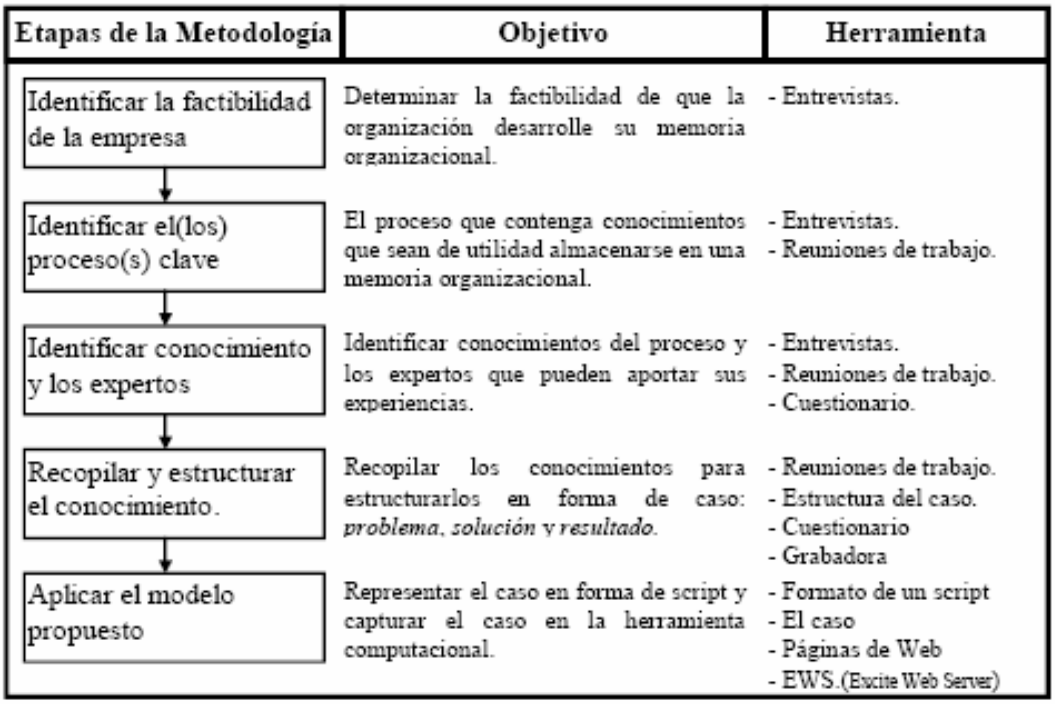


Fig. 5.1. Metodología para Desarrollar Memorias Organizacionales en Base a Casos.

A continuación se describe la implementación de esta metodología para nuestro caso de estudio:

5.2.1.1 Identificar la factibilidad de que la empresa desarrolle su memoria organizacional.

En esta etapa, se realizó una entrevista con los directivos de la organización y se revisaron los puntos propuestos en la metodología.

1. **La alta dirección debe tener un interés por el desarrollo de una memoria organizacional.** Los directivos de la empresa expresaron su interés en desarrollar una memoria organizacional basada en casos, que les permita conservar las experiencias y conocimientos de los empleados, además de que puedan ser compartidos entre ellos mismos.

2. **La organización debe tener conocimientos documentados, en expertos, en personal de mucha experiencia y que desea compartir entre todos sus miembros**. La empresa cuenta con personal muy experimentado, así como algunos de sus conocimientos están documentados.

3. **La empresa cuenta con personal con deseos de adquirir y compartir sus conocimientos.** El personal que labora en esta organización continuamente está aprendiendo nuevas y mejores formas de hacer las cosas, comparten su conocimiento y se ayudan en el trabajo diario.

De acuerdo a todo lo anterior, se consideró que la empresa es factible para que desarrolle su memoria organizacional.

5.2.1.2 Identificar el proceso clave de la empresa.

La empresa cuenta con diferentes procesos, se decidió por el proceso de implementación de sistemas ERP por ser la principal actividad de la Empresa.

1. **Debe ser un proceso clave para la organización.** La empresa lo considera como clave, ya que es el principal proceso de la empresa.
2. **Tiene impacto y da un valor agregado a la empresa.** Este es uno de los procesos de mayor impacto en la empresa ya que es la actividad principal de la empresa.
3. **Permite satisfacer los requerimientos del cliente.** El producto final de este proceso, consiste en que los requerimientos o consulta de lo consultores y jefes de proyecto podrán ser resueltas rápida y eficientemente.
4. **Balancea de forma efectiva los recursos humanos y tecnológicos de la empresa.** Este proceso debe tener alta precisión, requiere de mucha habilidad y conocimientos por parte de los empleados. La interacción de estos factores hace que este proceso requiera de una combinación efectiva de los recursos humanos y tecnológicos.

Por estas razones, se consideró que el proceso clave de la empresa es la implementación del ERP. Este proceso consiste básicamente en cumplir con la estructura y los tiempos estimados para algún proyecto especifico en el cual se desea implementar el ERP.

Los pasos que se siguen para la implementación son los siguientes:

1. **Definición:** el equipo de gestión de proyecto planea el proyecto de implantación. Los objetivos son identificar requisitos de negocio y de sistema, proponer el modelo comercial futuro, y proponer la aplicación y la arquitectura de tecnología de la información. El equipo desarrolla un plan educativo para asegurar que los miembros del equipo reciban entrenamiento y soporte necesario para realizar sus papeles en el proyecto.
2. **Análisis Operacional:** el equipo de proyecto recoge información sobre los procesos de negocio del usuario final y sus requerimientos. El equipo de proyecto desarrolla escenarios de los requerimientos del usuario para evaluar el nivel ideal entre los requisitos comerciales detallados y la funcionalidad estándar de aplicación.
3. **Diseño Solución:** El propósito de esta fase es crear la solución de procesos de negocio óptima para encontrar los requerimientos futuros. Durante esta fase, los miembros del equipo de proyecto diseñan opciones de configuración de la aplicación y documentan detalladamente los procesos de negocio.

4. **Construcción:** Durante la fase de construcción, el equipo de desarrollo codifica y prueba todas las extensiones personalizadas incluyendo mejoras de la aplicación, conversiones, e interfaces. El equipo crea y ejecuta test scripts de desempeño, integración y funcionalidad del sistema.

5. **Transición:** Durante la transición, el equipo de proyecto despliega la aplicación terminada en la organización, ejecuta la conversión de datos y usa la documentación desarrollada para entrenar a los usuarios finales y al equipo de mantenimiento.

6. **Producción:** El objetivo de esta fase es proveer al cliente el soporte necesario para el resto de vida del sistema.

5.2.1.3 Identificar el conocimiento y los expertos

Después de haber seleccionado el proceso clave de la empresa, la tercera etapa de la metodología consiste en identificar el conocimiento que sea útil documentar en una memoria organizacional e identificar a los expertos que participarán en el proceso de la obtención de los casos.

1. **Hacer un listado de aquellas situaciones pasadas que han dejado experiencias útiles.** Se realizó una entrevista con los consultores y jefes de proyecto que son los responsables del proceso de implementación del ERP y se le explicó las características que deberían tener las situaciones pasadas para que fueran factibles documentarlas en forma de caso.

2. **Seleccionar a la(s) persona(s) de mayor experiencia que ha(n) participado directamente en estas situaciones para que aporte su experiencia.** Después de seleccionadas las situaciones pasadas, se le preguntó a los responsable de los procesos quién o quienes son las personas de mayor experiencia y que hayan participado en esa situación, para el caso.

3. **Identificar el conocimiento útil de las situaciones pasadas**: Se realizó una descripción detallada de problemas y sus soluciones por parte de los expertos; habilidades o capacidades que han desarrollado los miembros más antiguos al realizar su trabajo; estrategias o ideas que se han aplicado en situaciones difíciles, cómo le han hecho, qué resultados han tenido; historias de éxito y fracaso.

5.2.1.4 Recopilar y estructurar el conocimiento en forma de caso.

Después de identificado el conocimiento y los expertos, la cuarta etapa de la metodología consiste en recopilar y estructurar el conocimiento en forma de caso.

1. **Recopilar el conocimiento del experto.** Para obtener el conocimiento relevante de cada una de las experiencias pasadas, se llevó a cabo una entrevista con algunos consultores y jefes de proyecto, así como también se consultaron algunos documentos elaborados sobre algunos temas.

2. **Estructurar el conocimiento en forma de caso.** Después de las entrevistas y la revisión de los documentos encontrados, se comenzó a escribir en papel todos los puntos importantes del problema, solución y resultado de la situación pasada.

3. **Verificar el contenido del caso con el experto.** Para asegurarse de la veracidad del contenido del caso, este fue revisado por los consultores y jefes de proyecto quienes leyeron los casos escritos y se hizo las correcciones y adecuaciones necesarias. Una vez verificado por el experto, se pudo pasar a su captura en la herramienta computacional.

5.2.1.5 Validar el Modelo Propuesto.

Después de recopilado y estructurado el conocimiento en forma de caso, la quinta etapa de la metodología consiste en validar el modelo propuesto. Para lograr esto, se debe representar el caso en forma de script, capturar el caso y el script en la herramienta computacional y utilizarla para verificar su funcionalidad.

1. **Representar el caso en forma de script.** Representar el caso siguiendo el formato de representación de un script. Una vez verificado el contenido del caso, se representó en un script. Para llevar a cabo esta representación, se leyó detenidamente el caso y se seleccionaron los eventos más importantes del problema, solución y resultado.

2. **Capturar el caso en la herramienta computacional.** Capturar el caso como parte de la base de conocimientos de la herramienta jColibri para que este disponible para su consulta. Una vez verificado el contenido del caso, este se capturó en la base de datos, se le dio la presentación apropiada de acuerdo a la estructura de la Base de Datos.

3. **Utilizar la herramienta de memoria organizacional para verificar su funcionalidad.** Plantear casos hipotéticos. Una vez capturados todos los casos en la herramienta, se plantearon casos hipotéticos de un nuevo problema. El objetivo era verificar que dado un nuevo problema, la herramienta computacional fuera capaz de mostrar casos similares que pudieran apoyar a la solución del problema. Esta verificación se realizara en los trabajos futuros ya que este trabajo plantea el diseño de las interfases de la herramienta, más no su desarrollo.

A continuación se Muestran dos casos representados en forma de Script.

Script Proveedor no Disponible en Ingreso de Facturas

Nombre:	Proveedor no Disponible en Ingreso de Facturas
Tipo de Caso:	Funcional
Modulo:	Cuentas por Pagar
Temática:	Facturas por Pagar
Nivel de Riesgo:	Alto
Método de Evaluación:	(NO Aplica)
Acción Tomada:	Aceptada

Escena 1: Problema
Al Registrar o Ingresar una Factura, un determinado proveedor no esta en lista de valores a pesar de que existe en el maestro de proveedores del sistema.

Escena 2: Solución
Se debe registrar la sucursal del proveedor, a pesar de que un proveedor exista en el maestro, debe tener registrada una sucursal en la organización actual para que este se pueda mostrar en la lista de valores de proveedores desde la pantalla de Ingreso de facturas.

Escena 3: Resultado
El proveedor en cuestión ya se encuentra disponible como dato de entrada en la lista de valores para ingresarlo en la factura.

Fig. 5.2. Representación en Script de un Caso Tipo.

Script Aparición de Nuevos Requerimientos que Pueden Alterar los Tiempos Planificados para el Proyecto.

Nombre:	Aparición de Nuevos Requerimientos que Pueden Alterar los Tiempos Planificados para el Proyecto
Tipo de Caso:	Gestión
Modulo:	Gestión
Temática:	Gestión de Tiempos del Proyecto
Nivel de Riesgo:	Alto
Método de Evaluación:	Directorial
Acción Tomada:	Aceptada

Escena 1: Problema
Se tienen nuevas definiciones o requerimientos no contemplados en el alcance de proyecto o contrato inicial.

Escena 2: Solución
Estos nuevos requerimientos que requieren cambios en la planificación del cronograma del proyecto serán tomados como adicionales a cotizar para el cliente luego del proyecto, ya que no estaban considerados dentro del contrato inicial por lo cual no se considero una planificación para su desarrollo en estos periodos.

Escena 3: Resultado
La decisión tomada favorece el flujo del cronograma en su estructura planificada ya que no se realiza ninguna modificación a el y todo sigue de acuerdo a lo planificado en gestión de tiempo.

Fig. 5.3. Representación en Script de Otro Caso Tipo.

5.3 Modelado del Negocio:

Para el modelado se ha utilizado el lenguaje UML.

5.3.1 Actores del Sistema

A continuación se describen los Actores del Sistema, así como tanbien las principales características o requisitos de cada uno de ellos:

5.3.1.1 Usuario

Los trabajadores de la Empresa HP S.A.C. que serán usuarios de la Herramienta son los que llevan las funciones de Consultores y los Jefes de Proyectos. A continuación se describa las funciones y características de cada tipo de usuario:

5.3.1.1.1 Consultor

Personal de la empresa que trabaja a tiempo completo en el proyecto. En este proceso de implementación de sistemas ERP, realiza las tareas de recopilar información, prepararla, ayuda en la toma de decisiones, etc. Trabajará en la implementación del modulo seleccionado.

Algunas de sus tareas consisten en.

1. Formar a los usuarios en el uso de la herramienta.
2. Realizar el análisis inicial de la empresa.
3. Realizar tomas de requerimientos de funcionalidad no contemplada por la herramienta.
4. Buscar soluciones, sin desarrollos adicionales.
5. Vencer las posibles reticencias de los usuarios al cambio.
6. Ayudar al Cliente en la definición de los nuevos procesos de negocio.
7. Mitigar riesgos que puedan surgir poniendo en peligro la implantación.

5.3.1.1.2 Jefe de Proyectos

El Jefe de Proyecto se destaca como la figura clave en la planificación, ejecución y control del proyecto y es el motor que ha de impulsar el avance del mismo mediante la toma de decisiones tendentes a la consecución de los objetivos.

El Jefe de Proyecto es un verdadero jefe, es decir, tiene poder ejecutivo y autoridad para mandar y tomar decisiones dentro del ámbito y objetivos del proyecto. No es un mero coordinador o animador, como en algunas ocasiones se piensa. De la misma forma, tampoco sería correcto pensar que el Jefe de Proyecto tiene un poder absoluto y dictatorial sobre el mismo, ya que se encuentra inmerso en la estructura y organización de la empresa.

Jefe de proyecto es el jefe técnico del proyecto para la etapa de puesta en marcha y sus funciones son la de acordar con el cliente los cronogramas del proyecto, además de ser el responsable de Implantación, Mantenimiento, Operación,

Sistemas, Seguridad y Calidad, asegurándola entrega conforme de todos los servicios al Cliente.

Entre otras, sus labores serán las que siguen:

1. Cotizar implantaciones o realizar presupuestos de implantación.
2. Planificar y definir las fases, recursos y tiempo requerido para la implantación.
3. Realizar el seguimiento de costes y desviaciones del proyecto
4. Formar a nuevos consultores.
5. Asignar tareas y realizar el seguimiento a los consultores de su equipo.
6. Realizar el seguimiento de la satisfacción del Cliente en cuanto a la implantación.
7. Realizar los informes de seguimiento del proyecto
8. Planificar y dirigir las reuniones de seguimiento del proyecto.

5.3.1.2 Experto

Los trabajadores de la Empresa HP S.A.C. que han sido catalogados dentro de este grupo son los que cumplen con los siguientes requisitos:

5.3.1.2.1 Perfil Para los Consultores Expertos:

Al menos 3 años de experiencia como Consultor en Proyectos de este Tipo.

Sólidos conocimientos técnicos y generales de los Módulos Financieros.

Orientación de Servicio al cliente.

5.3.1.2.2 Perfil Para los Jefes de Proyectos Expertos:

Al menos 3 años de experiencia como Jefe de Proyectos.

Debe dominar la tecnología principal del proyecto, planificar los recursos, generar ideas y soluciones eficaces, controlar la calidad, etc.

Debe poseer una capacidad destacada para las relaciones personales, puesto por un lado, es el representante principal del proyecto ante clientes, proveedores, subcontratistas, otras direcciones funcionales, la propia empresa, y por otro, debe dirigir a un conjunto de personas sobre los que normalmente no tiene poder jerárquico, y por lo tanto, es necesario hacerlo con grandes dosis de autoridad personal, tacto, habilidad y capacidad de convicción.

Debe poseer una notable aptitud gestora, pues no sólo se encarga de una dimensión técnica, sino que debe controlar y conseguir todos los objetivos del proyecto, incluyendo los financieros y de plazo, que suelen ser los más críticos y más frecuentemente incumplidos.

5.3.2 Diagrama de Casos de Uso

La Herramienta de Razonamiento Basado en Casos para toma de decisiones en proyectos de implementación de Sistemas ERP, presenta el siguiente caso de uso de Negocio:

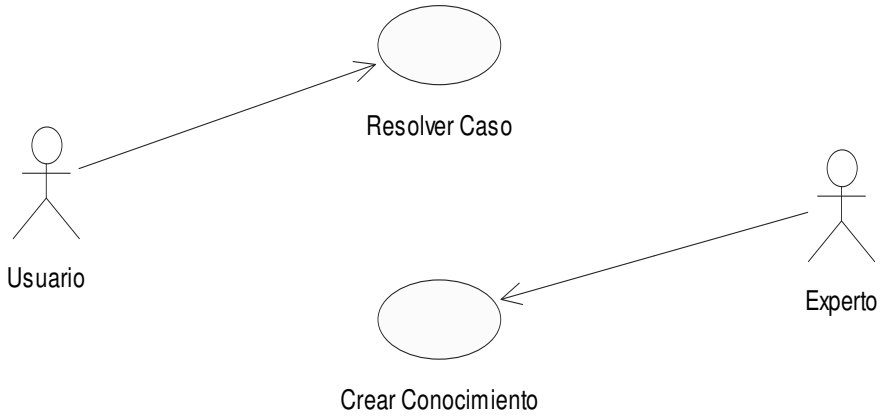


Fig. 5.4. Diagrama de Casos de Uso.

5.3.3 Descripción de los Casos de Uso:

Caso de Uso: Resolver Caso
Objetivo: Brindar apoyo en la Solución de los Nuevos Casos
Actores: Usuario
Condiciones Previas: • El Usuario debe haber hecho una consulta.
Flujo Principal: 1. El usuario ingresa un caso nuevo al sistema. 2. El sistema recibe la consulta. 3. El sistema recupera los casos similares almacenados. 4. El sistema compara y revisa las soluciones anteriores. 5. El sistema adapta una solución nueva al caso 6. Se envía la posible solución al caso nuevo.
Variantes o extensiones: 1. Si el sistema no encuentra un caso similar. 2. Guarda la consulta en la base, y se resuelve el problema manualmente. 3. El experto almacenara el nuevo conocimiento en la base de casos.

Tabla 5.1. Descripción del Caso de Uso: Resolver Caso

Caso de Uso: Ingresar Conocimiento
Objetivo: El sistema tenga conocimiento almacenado para poder resolver casos nuevos.
Actores: Experto
Condiciones Previas: • Ninguno.
Flujo Principal: 1. El usuario ingresa al sistema. 2. El experto clasifica los casos nuevos a ingresarse. 3. El experto revisa los casos que se podrían ingresar. 4. El experto ingresa el conocimiento. 5. El sistema almacena el nuevo conocimiento en la base de casos

Tabla 5.2. Descripción del Caso de Uso: Ingresar Conocimiento

5.3.4 Diagrama de Actividades

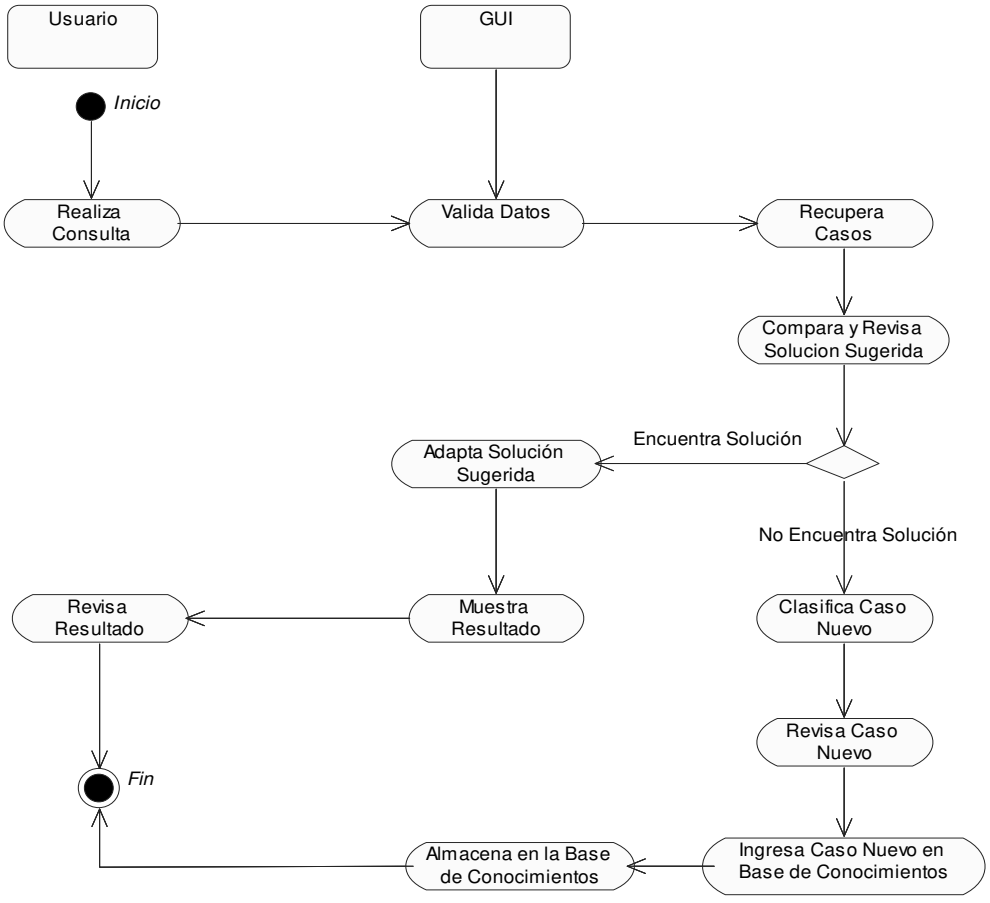


Fig. 5.5. Diagrama de Actividades.

5.4 Diseño:

5.4.1 Diagrama de Clases:

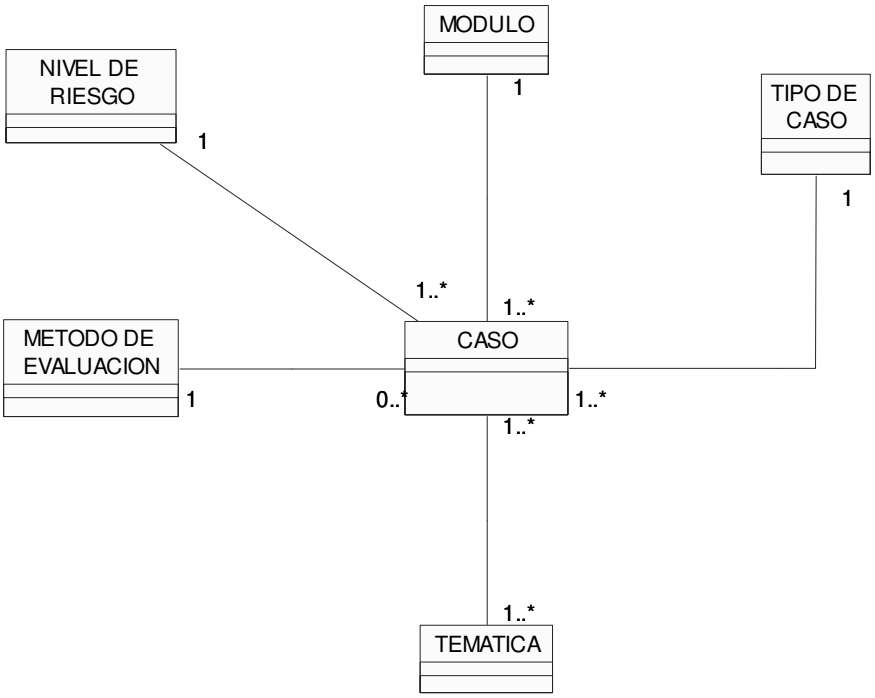


Fig. 5.6. Diagrama de Clases.

5.4.2 Diagramas de Secuencia:

5.4.2.1 Resolver Caso:

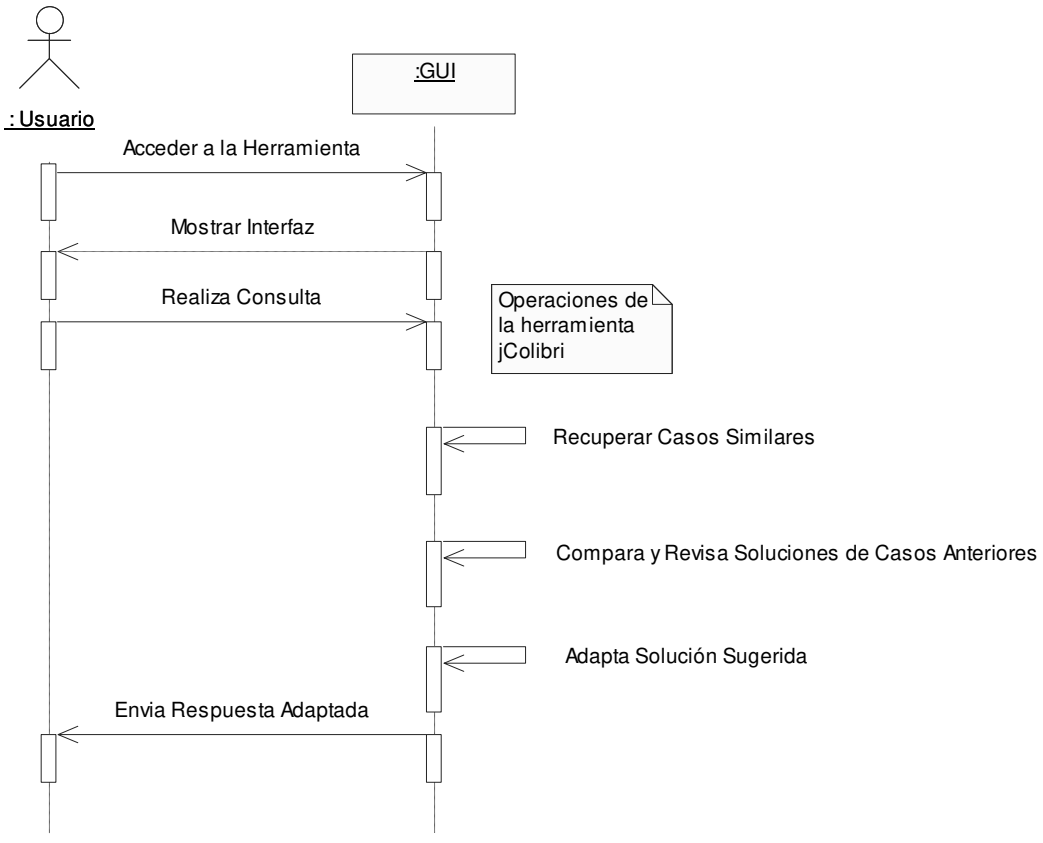


Fig. 5.7. Diagrama de Secuencia: Resolver Caso.

5.4.2.2 Crear Conocimiento:

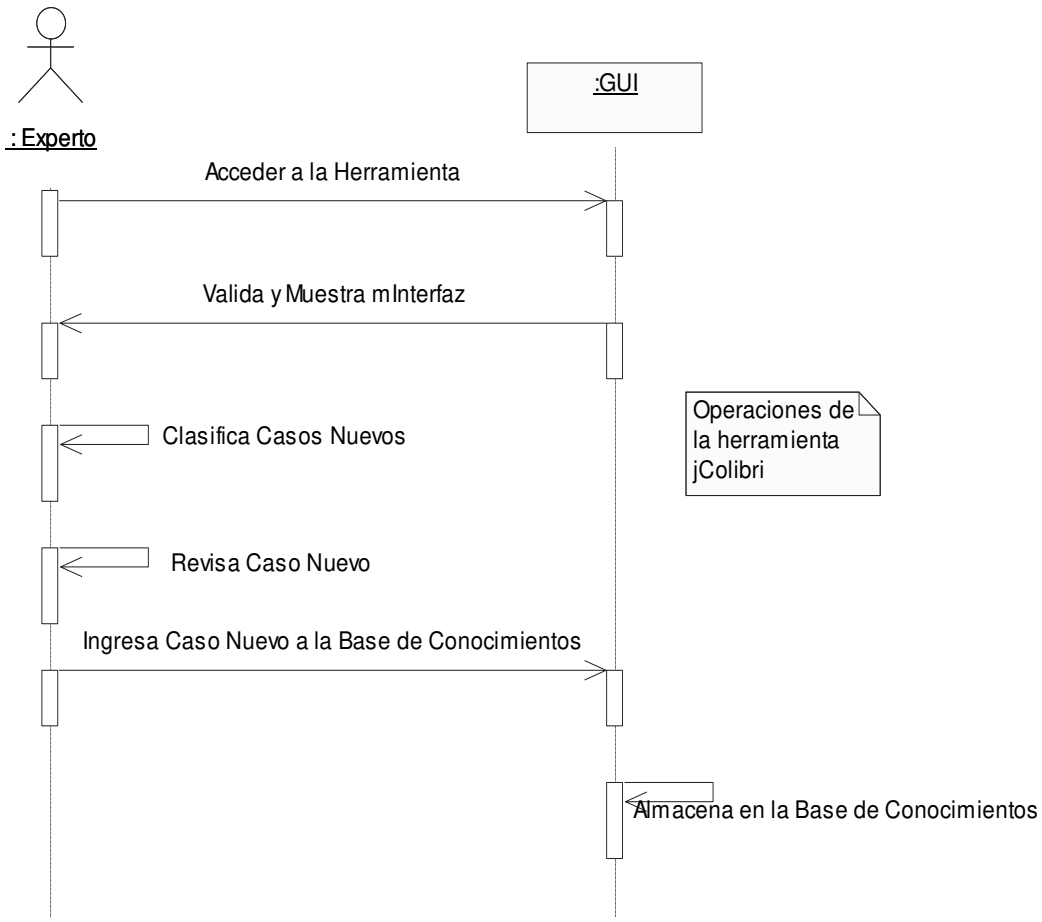


Fig. 5.8. Diagrama de Secuencia: Crear Conocimiento.

5.4.3 Diagramas de Colaboración:

5.4.3.1 Resolver Caso:

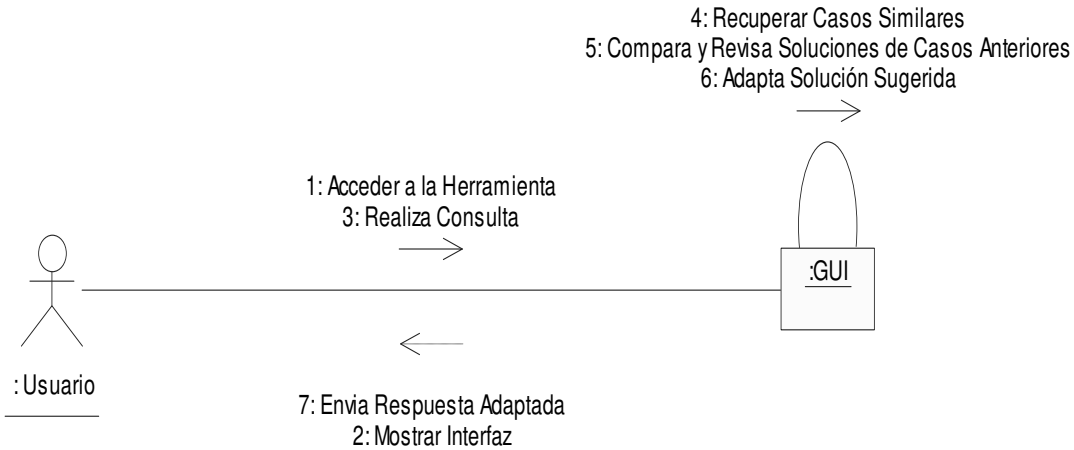


Fig. 5.9. Diagrama de Colaboración: Resolver Caso.

5.4.3.2 Crear Conocimiento:

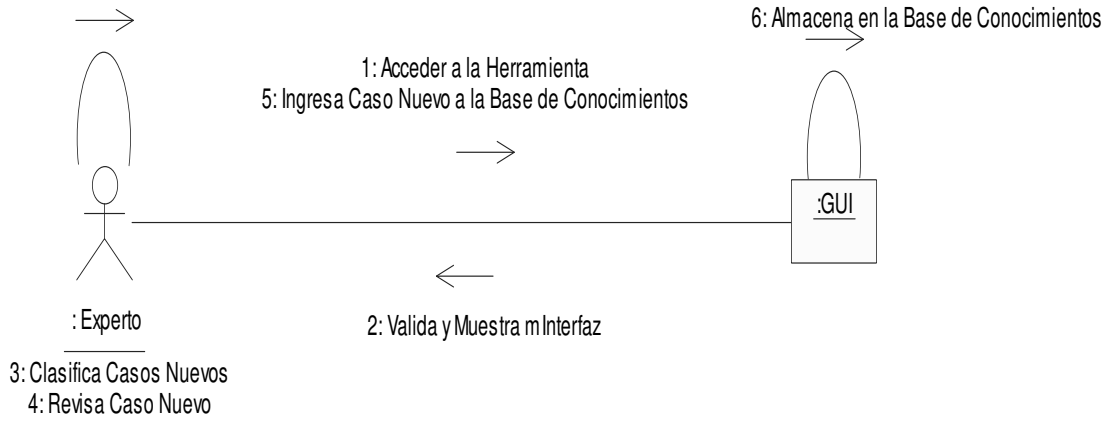


Fig. 5.10. Diagrama de Colaboración: Crear Conocimiento.

5.5 Prototipos

5.5.1 Ingresar Consulta

Esta pantalla permitirá al usuario consultor ingresar atributos de acuerdo al tipo de caso presentado (funcional o de gestión) y una breve descripción del problema. Luego, presionando el botón Consultar el sistema mostrará el caso más parecido al ingresado.

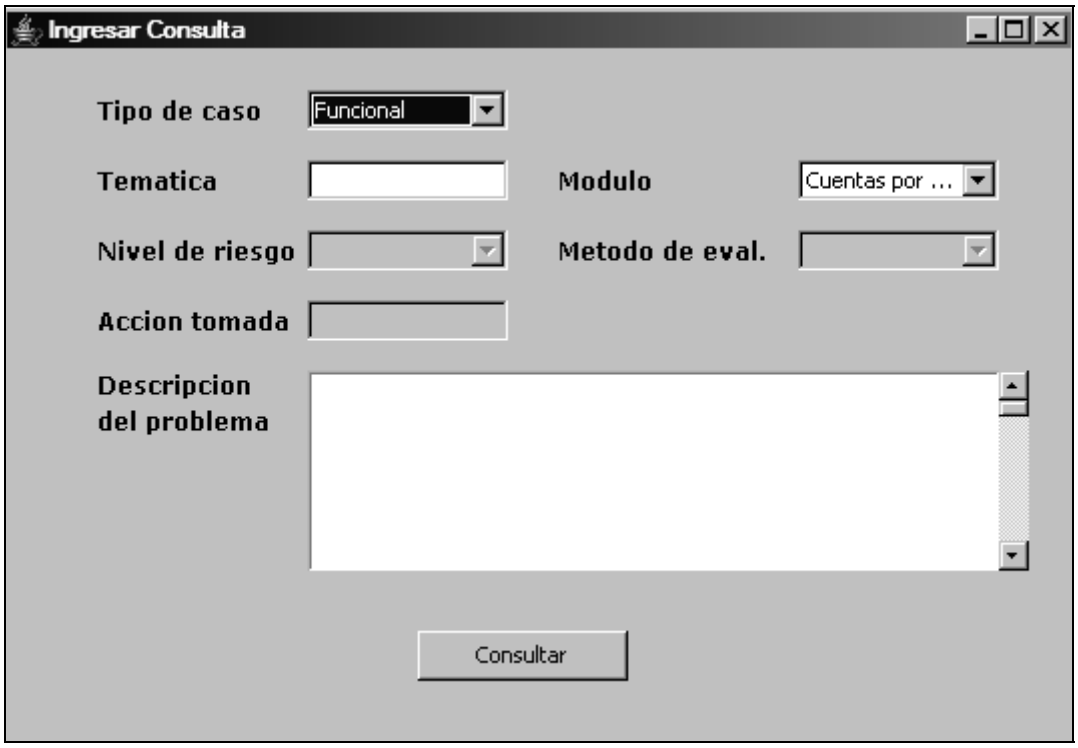


Fig. 5.11. Prototipo de la Pantalla de Ingresar Consulta.

5.5.2 Resultado de Consulta

Esta pantalla muestra el resultado de la consulta ingresada en la pantalla anterior.

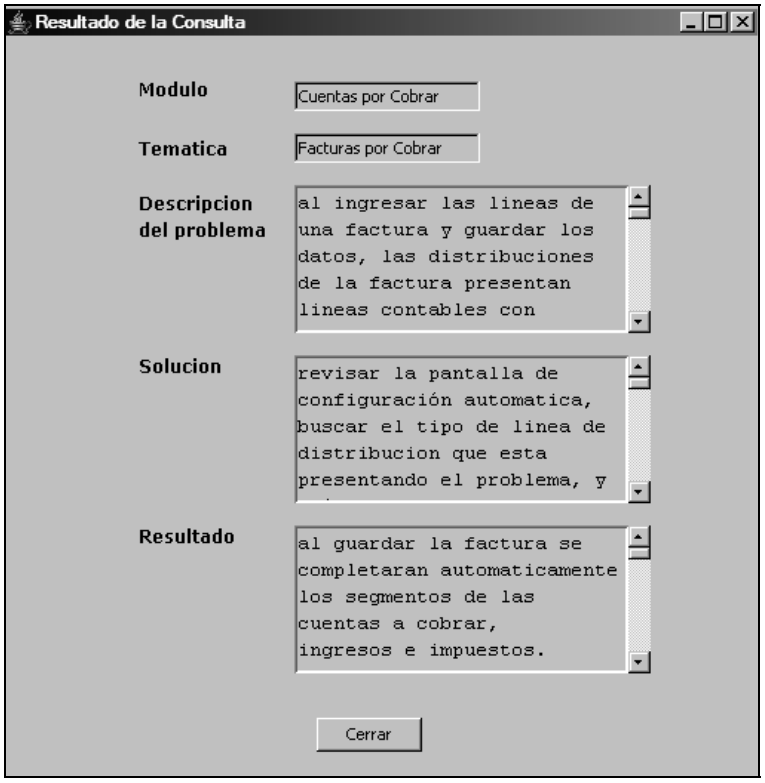


Fig. 5.12. Prototipo de Pantalla de Resultado.

5.5.3 Registrar Caso

Esta pantalla permite registrar nuevos casos en la base de casos.

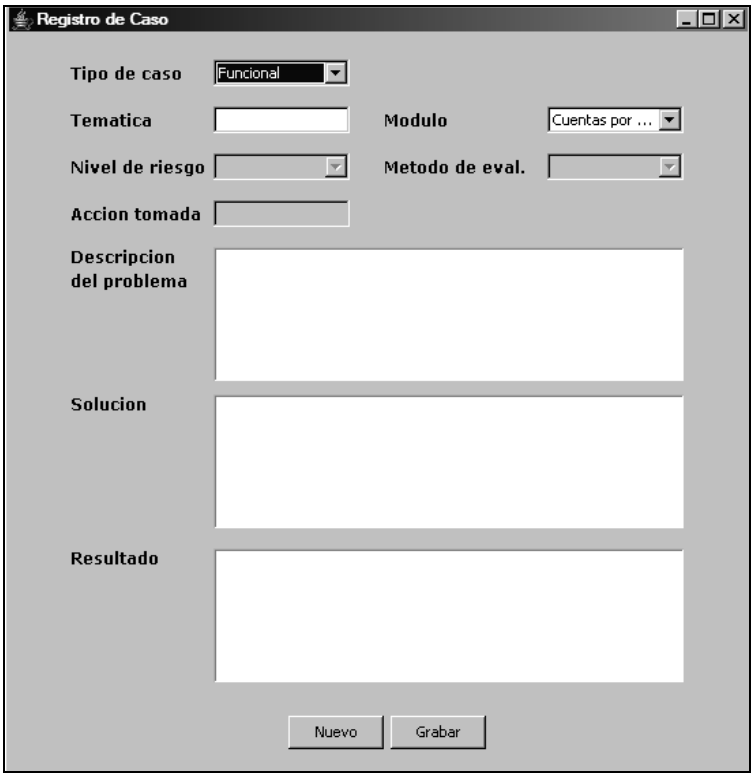


Fig. 5.13. Prototipo de Pantalla ingresar Caso.

5.6 Tareas Configuradas en jColibri

En la figura 5.14., se muestra la representación la lista de tareas configuradas en la herramienta jColibri que deben configurarse en el jColibri para este caso de estudio.

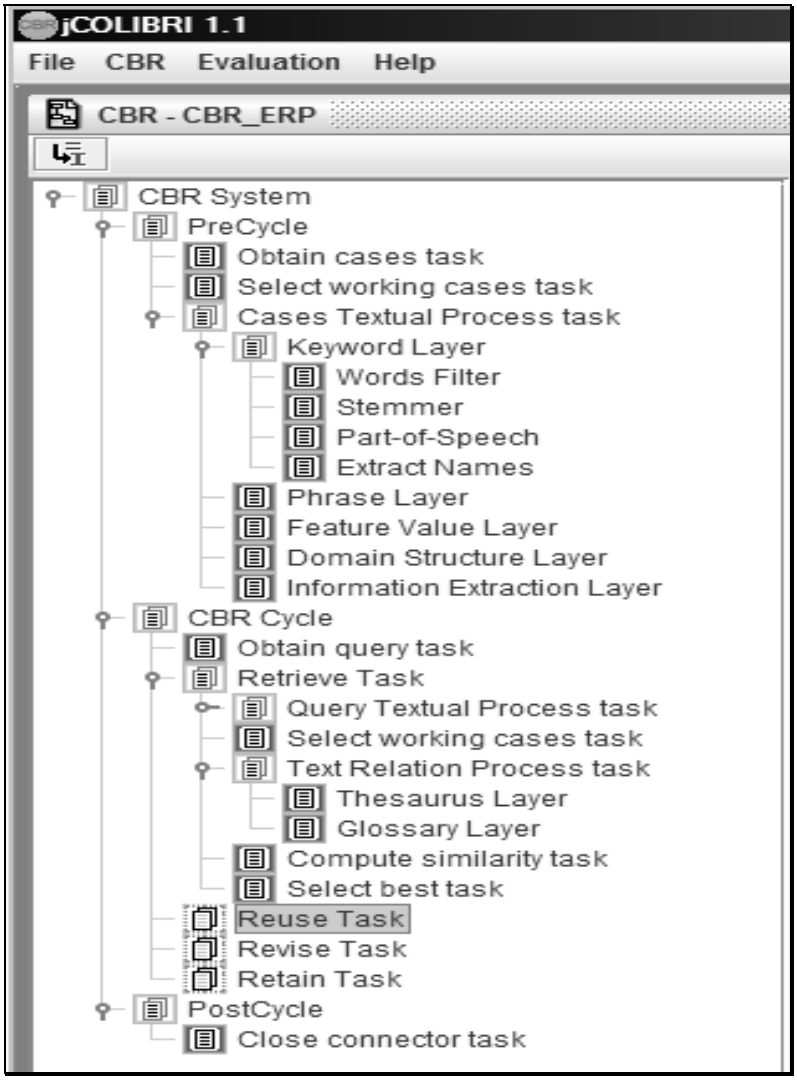


Fig. 5.14. Lista de Tareas configuradas en jColibri.

En esta Figura, se puede observar las 4 fases principales del ciclo CBR de las cuales se ha configurado 2 de ellas.

La herramienta implementa el modelo de Lenz [75] (descrito en el capítulo 2) así como sus fases principales.

En el Pre-Ciclo se ejecutan las tareas para cargar los casos almacenados en memoria.

En el Ciclo se realiza nuevamente este proceso dentro de la tarea “Retrieve Task” para extraer la información de la consulta ingresada por el usuario.

Posteriormente se realizan el resto de tareas del ciclo.

Finalmente se ejecuta el post-ciclo donde se cierra la conexión con la base de casos y se libera la memoria.

5.7 Requerimientos Mínimos de Hardware y Software:

Para la implementación del modelo en el proceso real, será necesario desarrollar una aplicación en Java que permita guardar en una Base de Datos, los datos de los casos y las variables respectivas empleadas en el proceso de toma de decisiones en proyectos de implementación de sistemas ERP; e integrarlo con el proyecto generado por jColibri. Luego el sistema será instalado en cada una de las computadoras utilizadas por los consultores y jefes de proyecto que laboran en la consultora HSP S.A.C., con una BD centralizada que maneje la persistencia.

En cuanto al Hardware necesario para el servidor de Base de Datos, se recomienda que la base de datos central este en un Servidor independiente y que tenga la mayor capacidad posible para los equipos de este tipo, debido al manejo abundante de información.

Los requerimientos mínimos de Hardware y Software para el desarrollo del sistema:

Hardware:

- Procesador Pentium IV, 2GHz a más.
- Memoria RAM 2GB mínimo.
- Disco Duro: 200GB o mayor dependiendo del tamaño de la base de casos.

Software:

- Sistema Operativo Windows XP o superior.
- Manejador de Base de Datos MySQL 4.1 Versión Server o superior.
- Framework jColibri.
- IDE Eclipse for Java Developers.

Para la máquina cliente del usuario del sistema:

Hardware:

- Computadora Pentium IV, a más.
- Memoria RAM 1GB mínimo.

Software:

- Sistema Operativo Windows XP o superior.
- Manejador de Base de Datos MySQL 4.1 Versión Cliente.
- Instalar jdk 1.6 o versión superior.
- Aplicativo generado en jColibri.

5.8 Validación y Análisis de resultados:

De acuerdo al alcance del presente trabajo se realizó el análisis del sistema que implementará dicho modelo, más no el desarrollo propio de este en un ambiente de producción simulado. Debido a esta razón, para validar el modelo y analizar los resultados obtenidos de su implementación se debe realizar dichas tareas que ofrezcan resultados más cercanos a la realidad.

A pesar de lo expuesto en el párrafo anterior, se han realizado pruebas simples de la funcionalidad del modelo implementado en la herramienta jCOLIBRI. Estas pruebas consistieron en un flujo simple de trabajo, donde el usuario consultor ingresa los valores de los atributos del problema que quiere resolver, el sistema mostró la similitud de la consulta ingresada con cada uno de los casos existentes en la base de casos. Al final se muestra cual es el caso que tiene mayor similitud, el cual puede ser tomado como solución.

La realización de estas pruebas reflejan un indicador de mejora de tiempo en la resolución de problemas en los proyectos de implementación, sobre todo para consultores con poca experiencia, que antes tenían que invertir mayor tiempo en investigar o revisar la documentación de la aplicación. Adicionalmente para consultores con mayor experiencia, que muchas veces tiene que recurrir a su memoria, las soluciones propuestas ayudan a conseguir y enriquecer aun más la información en caso el usuario pueda añadir conocimiento a dichos casos.

CAPITULO VI

CONCLUSIONES Y TRABAJOS FUTUROS

6.1 Conclusiones

- La implementación del modelo propuesto apoyará a las empresas encargadas de implementar los sistemas ERP dado que se cuenta con el conocimiento oportuno para responder rápidamente ante una toma de decisión que pudiera presentarse, con lo cual se estaría garantizando la calidad de la información con la que se cuente y las decisiones que se tomen, así como la reducción de esfuerzo y tiempo.

- Se diseñó una estructura de la base de casos que permita manejar la información de acuerdo a los objetivos trazados, también se estableció una Ontología para enriquecer la gestión de dicho conocimiento dentro de la aplicación, logrando así modelar el conocimiento de la Organización.

- El Modelo propuesto permite resaltar la importancia de la gestión conocimiento dentro de la organización, creando una cultura organizacional alineada a esta buena práctica.

- Se logró adquirir y gestionar el conocimiento dentro de la organización a través del modelo propuesto, lo cual permitirá su reutilización eficiente.

- Con la implementación de esta aplicación se logra extraer la información fácilmente basándose en experiencias pasadas, apoyada en la técnica de Razonamiento Basado en Casos.

6.2 Trabajos futuros

- El alcance de este trabajo abarcó el análisis y el desarrollo de prototipos para la solución, por lo cual, se recomienda una implementación total del sistema ampliando sus funcionalidad y mejorando su expresividad y presentación.

- Utilizar la segunda versión de jCOLIBRI, la cual presenta muchas mejoras respecto a su antecesora, con lo cual, las aplicaciones obtenidas serán muchos más eficientes y tendrán resultados más satisfactorios.

- Utilizar otras herramientas que faciliten la construcción de sistemas RBC, y hacer un análisis profundo y comparativo de tiempo, costo y beneficio de las aplicaciones resultantes, para poder determinar cuál de ellas es la más recomendable al momento de diseñar este tipo de aplicaciones.

- Se definió una ontología para el caso, la ontología que se creó para esta aplicación recopila información aplicable a la toma de decisiones en los proyectos de implementación de sistemas ERP basándose en las experiencias anteriores la cual puede ser mejorada y ampliada a más módulos y/o temáticas a tratar, ya que el conocimiento inmerso en los Sistemas ERP es muy grande.

Referencias Bibliográficas

[1].- Gonzalo Villarreal Farah, Inteligencia Artificial: Sistemas Expertos. Universidad de Santiago de Chile.
http://www.comenius.usach.cl/gvillarr/cursoia/present/Sis-Exp%20P2.ppt

[2].- José Antonio San miguel Carrillo, Introducción al Razonamiento Basado en Casos, Universidad de Valladolid.
http://www.infor.uva.es/~calonso/IAII/Aprendizaje/TrabajoAlumnos/RBCmemoria.pdf

[3].- Patricio Ramírez Correa, "Rol y Contribución de los Sistemas de planificación de los Recursos Empresariales (ERP)", Universidad de Sevilla, Tesis Doctoral (2004), Sevilla - España

[4].- Julio César Dueñas Bustinza, Sistemas de Gestión basados en ERP, Universidad Andina de Cuzco,
http://www.uandina.edu.pe/download/is/erp.pdf

[5]. - ORACLE Corporation, AIM: A Comprehensive Method and Toolkit for Implementing Oracle's Packaged Applications.
http://www.oracle.com/consulting/collateral/AIMadvantage.pdf

[6].- Castillo E.,Gutierrez J.M., Hadi A.S., "Sistemas Expertos y Modelos de Redes Probabilísticas", Academia Española de Ingeniería, (1998), Madrid – España

[7].- Rossillea D., Laurentc J., Burguna A., "Modelling a Decision-Support System for Oncology using Rule-Based and Case-Based Reasoning Methodologies", International Journal of Medical Informatics Vol 74 (2005) 2-4

[8]. - Reyes R., Sison R., "Case Retrieval in CBR-Tutor, Proceeding of the International Conference on Computer in Education" – ICCE´02 IEEE (2002)

[9]. - Aamodt A., Plaza E., "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", Artificial Intelligence Communications 7, Vol 1 (1994) 39-52.

[10]. - Rashid M. A., Hossain L., Patrick J. D. (Eds.), "Enterprise Resource Planning: Global Opportunities and Challenges", Idea Group Publishing, (2002), Hershey.

[11]. - Davenport T., "Putting the Enterprise into the Enterprise System", Harvard Business Review, Vol 76 (1998) 121-131.

[12]. - Markus M., Tanis C., "The enterprise systems experience – from adoption to success, Framing the Domains of IT Research: Glimpsing the Future through the Past", Zmud, Pinnaflex Educational Resources, Cincinnati, (2000) 173–207.

[13]. - Parr A., Shanks G, "A Model of ERP Project Implementation", Journal of Information Technology, Vol 15 (2000) 289-304.

[14]. - Ahituv N., Neumann S., Zviran M., "A system development methodology for ERP systems", Journal of Computer Information Systems, Vol 42 (2002) 56-67.

[15]. - Gamma E., Helm R., Johnson R., Vlissides J., “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley Professional Computing Series, (1994)

[16]. - Gruber T. R., “A translation approach to portable ontologies”, Knowledge Acquisition, (1993)

[17]. - Federico Peinado Gil, “Un armazón para el desarrollo de aplicaciones de narración automática basado en componentes ontológicos reutilizables”, Universidad Complutense de Madrid, Tesis Doctoral (2008), Madrid - España

[18]. - Fernando Sáez Vacas, Oscar García, Javier Palao, Pedro Rojo, “Innovación Tecnológica en las Empresas. Temas Básicos”, Universidad Politécnica de Madrid, ETS de Ingenieros de Telecomunicación, (2003), Madrid – España.

[19]. – Alejandro Andrés Pavez Salazar, “Modelo de Implantación de Gestión del Conocimiento y Tecnologías de Información para la Generación de Ventajas Competitivas”, Universidad Técnica Federico Santa María, Tesis para optar Titulo (2000), Valparaíso – Chile.

[20]. - A. T. Schreiber, B. J. Wielinga, and e. J. A. Breuker, “KADS: A Principled Approach to Knowledge-Based System Development”, Volume 11 of Knowledge-Based Systems Book Series, Academic Press, London, 1993.

[21]. - B. Díaz-Agudo,”Una aproximación ontológica al desarrollo de sistemas de razonamiento basado en casos”, Tesis doctoral, Universidad Complutense de Madrid, 2002.

[22]. - B. Díaz-Agudo y P.A. González-Calero,”CBROnto: a task/method ontology for CBR”, In S. Haller and G. Simmons, editors, Procs. Of the 15th International FLAIRS’02 Conference (Special Track in CBR, pages 101-106. AAAI Press, 2002.

[23]. - Página web del proyecto LOOM:
http://www.isi.edu/isd/LOOM/LOOMHOME.html

[24]. - Página web del proyecto RACER:
http://www.sts.tu-harburg.de/~r.f.moeller/racer/

[25]. – Antonio A. Sanchez Ruiz Grados, “jColibri: estado Actual y Posibles Mejoras”, Curso Aprendizaje Automatico, Universidad Complutense de Madrid, 2004-2005, Madrid –España.

[26]. – Juan Antonio Recio Garcia, “jColibri: Una Plataforma Multi-nivel para la construcción y generación de Sistemas de Razonamiento Basado en Casos”, Tesis Doctoral, Universidad Complutense de Madrid, 2008, Madrid – España.

[27]. – Juan A. Recio-García, Belén Díaz-Agudo, Marco A. Gómez-Martín, and Nirmalie Wiratunga, "Extending jCOLIBRI for Textual CBR", Proceedings of Case-Based Reasoning Research and Development, 6th International Conference on Case-Based Reasoning, ICCBR 2005, pages 421-435, Chicago, IL, US, Springer, August 2005.

[28]. – Pedro Pablo Gomez Martin, “Modelo de Enseñanza Basada en Casos: De los Tutores Inteligentes a los Video Juegos”, Tesis Doctoral, Universidad Complutense de Madrid, 2007, Madrid – España.

[29]. – Juan A. Recio-García, Belén Díaz-Agudo, Pedro Gonzales Calero, "jColibri Tutorial 2", Technical Report IT/2007/02, Universidad Complutense de Madrid, 2008, ISBN: 978-84-691-6204-0, Madrid – España.

[30]. - GAIA - Group for Artificial Intelligence Applications:
http://gaia.fdi.ucm.es/

[31]. – International Conference on Case-Based Reasoning:
http://www.iccbr.org/

[32]. – The Sixth International Conference on Case-Based Reasoning:
http://oucsace.cs.ohiou.edu/~marling/iccbr05/cfp.html

[33]. – Textual Case-Based Reasoning Workshop at the 6th International Conference on Case-Based Reasoning:
http://www.pages.drexel.edu/~rw37/tcbr05.html

[34]. – Departamento de Ciencias de la Computación de la Universidad de Auckland:
http://www.ai-cbr.org/

[35]. – SEPIA working Group:
http://decsai.ugr.es/~lcv/SEPIA/

[36]. – AI Magazine:
http://www.aaai.org/Magazine/magazine.php

[35]. – ARTIFICIAL INTELLIGENCE JOURNALS:
http://www.cs.iastate.edu/~honavar/aijournals.html

[36]. - Mendiz Noguero, Inés (2002) Un estudio sobre la aplicación del razonamiento basado en casos a la construcción de programas. Tesis Doctoral, Universidad Complutense de Madrid. España. Defendida con fecha 1992

[37]. - Díaz Agudo, María Belén (2004) Una aproximación ontológica al desarrollo de sistemas de razonamiento basado en casos. Tesis Doctoral, Universidad Complutense de Madrid. España. Defendida con fecha 2002

[38]. - Pous i Sabadí, Carles (2004) Case based reasoning as an extension of fault dictionary methods for linear electronic analog circuits diagnosis, Tesis Doctoral, Universidad de Girona

[39]. - Dra. Mercedes Medina Pagola, Utilización del Aprendizaje Basado en Problemas Bajo la Optica de la Inteligencia Artificial, Revista Cubana de Informática Médica, No. 1 Año 2 ISSN:1684-1859 , Año 2002.

[40]. - Martha D. Delgado Dapena, Definición del modelo del negocio y del dominio utilizando Razonamiento Basado en Casos, La Revista Electrónica del DIICC, ISSN: 0717 – 4195, Edición número 8 (26 de Agosto 2002)

[41]. - Publicación enviada por Dr. José Rodolfo Romero Villar y Dr. Fernando Sadulé Más, Inteligencia Artificial. Aplicaciones en medicina, Ilustrados.com, Publicado Tuesday 28 de March de 2006

[42]. - Gomez, S., Perichinsky, G. y García-Martínez, R., Argumentación Utilizando Razonamiento Basado en Precedentes en Sistemas Expertos Legales, Nuevas Tecnologías Informáticas Aplicadas al Derecho y a la Empresa (Capítulo), ISBN 950-31-0050-X, Pág. 65-74, 2001.

[43]. - Ghislain Atemezing y Juan Pavón, Intelligent Environment for Medical Practices in African Traditional Medicine, 6th International Workshop on Practical Applications on Agents and MultiAgent Systems. ISBN: 978-84-611-8858-1. Pág. 101-108, 2008.

[44]. – Contraloría General de la Republica, Manual del Sistema de Gestión del Conocimiento para el Control Gubernamental. Aportes para la Discusión, Contraloría General de la Republica. ISBN: 978-9972-854-46-0. 2008.

[45]. – P. Javier Herrera, P. Iglesias, D. Romero e I. Rubio, JaDaCook: Java Application Developed and Cooked Over Ontological Knowledge, Departamento de Ingeniería de Software e Inteligencia Artificial, Universidad Complutense de Madrid, Madrid – España, 2008.

[46]. - Ghislain Atemezing, Lauret Pozo y Juan Pavón, Ontology in Multi Agent Conception, Case of African Traditional Medicine, Departamento de Informática, Universidad de Yaounde, Yaounde - Camerun, 2008.

[47]. – Federico peinado Gil, Mediación Inteligente entre Autores e Interactores para Sistemas de narración Digital Interactiva, Tesis Doctoral, Facultad de Informática, Universidad Complutense de Madrid, Madrid - España, 2004.

[48]. – Felipe Romero, Construcción de Bases de Conocimiento para Apoyar la Estimación de Proyectos de Software por Analogía, Tesis, Facultad de Informática, Universidad de Los Andes, Colombia, 2008.

[49]. – Rafael Oswaldo Ruedas Casallas, Gestión del Conocimiento para la Reutilización de la Experiencia Obtenida en la Corrección de Defectos, Tesis de Postgrado, Facultad de Informática, Universidad de Los Andes, Colombia, 2008.

[50]. – Christian Indahl y Kjell Martin Rud, Arbitration and Planning of Workflow Processes in a Context – Rich Cooperative Environment, Tesis de Postgrado, Departamento de Ciencias de Computación e Informática, Universidad Noruega de Ciencias y Tecnología, Noruega, 2007.

[51]. – Walter Luis Mikos, Modelo Basado en Agentes como Solución a Problemas de No Conformidades en Ambientes de Manufactura con Recursos Distribuidos, Tesis de Postgrado, Facultad de Mecánica, Universidad Federal de Santa Catarina, Santa Catarina - Brasil, 2008.

[52]. – Hector Gómez, Belen Díaz – Agudo y Pedro Gonzáles Calero, Ontology – Driven Development of Conversational CBR Systems, Departamento de Sistemas Informáticos y Programacion, Universidad complutense de Madrid, Madrid - España, 2005.

[53]. – Héctor Gómez Gauchia, COBBER: un Enfoque Sistémico, Afectivo y Ontologico para el Razonamiento Basado en Casos Conversacional, Tesis Doctoral, Facultad de Informática, Universidad complutense de Madrid, Madrid - España, 2008.

[54]. – Héctor Gómez, Belén Díaz – Agudo y Pedro Gonzáles Calero, A Case Study of Structure Processing to Generate a Case Base, Departamento de Sistemas Informáticos y Programación, Universidad complutense de Madrid, Madrid - España, 2004.

[55]. – Antonio Sánchez-Ruiz, Stephen Lee-Urban, Héctor Muñoz-Ávila, Belén Díaz – Agudo y Pedro Gonzáles-Calero, Game AI for a Turn-based Strategy Game with Plan Adaptation and Ontology-based retrieval , Departamento de Sistemas Informáticos y Programación, Universidad complutense de Madrid, Madrid - España, 2004.

[56]. – Federico Peinado y Pablo Gervas, Creativity Usses in Plot Generation, Departamento de Sistemas Informáticos y Programación, Universidad complutense de Madrid, Madrid - España, 2004.

[57]. – Pérez Soltero, Alonso, "Modelo para la Representación de una Memoria Organizacional Utilizando Herramientas Computacionales de Internet", Tesis de Maestría, Instituto Tecnológico Y de Estudios Superiores de Monterrey, Monterrey - México, 1997.

[58]. - Riesbeck, Christopher. "Inside Case-Based Reasoning", Edit. Lawrence Erlbaum Associates, Inc., USA 1989.

[59]. - Barr, Avron, "The Handbook of Artificial Intelligence", William Kaufmann, Inc., USA 1981.

[60]. - Kolodner Janet, Case-Based Reasoning, edit. Morgan Kaufmann, USA 1993.

[61]. - Davenport Tom,"Knowledge Management", Presentación videograbada, Video No.1, Bloque No.3, a 2´14´´, duración 47´27". ITESM 1996.

[62]. - Molina Gutiérrez, Arturo. 1997, "Planeación e Integración tecnológica de empresas", Manual de Lecturas, Curso de Administración de la Innovación Tecnológica, ITESM, pp. 392-398.

[63]. - Alterman R, "Panel discussion on case representation", Proceedings of the Second Workshop on Case-Based Reasoning, Pensacola Beach, USA 1989.

[64]. – Pérez - Soltero, Alonso. "Memoria Organizacional Basada en Casos", Revista de Ciencia e Tecnología Política e Gestao para a Periferia (RECITEC), Vol. 6 No.1, pp. 22-39. 2002, ISSN 1415- 262, Recife, Brasil.

[65]. – G. van Heijst, A. T. Schreiber, y B. J. Wielinga, 1997, Using explicit ontologies in kbs development, International Journal of Human and Computer Studies, 46(2-3):183–292, ISSN 1071- 819.

[66]. – M. Riichiro, V. Johan, y I. Mitsuru, 1995, Task ontology for reuse of problem solving knowledge, Knowledge Building & Knowledge Sharing (KB&KS'95) (2nd International Conference on Very Large-Scale Knowledge Bases).

[67]. – B. Chandrasekaran, J. R. Josephson, y V. R. Benjamins, 1999, Ontologies: What are they? wy do we need them?, IEEE Intelligent Systems and their applications, 14(1):20–26.

[68]. – G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, y K. J. Miller, 1990. Introduction to ordNet: An On-line Lexical Database. International Journal of Lexicography, 3(4):235–244.

[69]. – Watson I, (1998). CBR is a methodology not a technology, In, Research & Development in Expert Systems XV, Miles, R., Moulton, M. & Bramer, M. (Eds.), pp. 213-223. Springer, London. ISBN 1-85233-086-4.

[70]. – Abdelmalik Moujahid, Iñaki Inza y Pedro Larrañaga, Clasificadores K-NN. Intelligent Systems Group, Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco. Gipuzkoa – España.

[71]. - Ana Juaristi Olalde, "Consejos y Pautas de un Consultor ERP para la elección de la mejor herramienta", Aula ERP. Gipuzkoa – España, 2009.

[72]. - Bearingpoint, "Estudio sobre Implantación de ERP en la Administración Publica", BearingPoint Business Consulting España, España, 2004.

[73]. - Rafael Oswaldo Rueda Casallas, "Gestión de conocimiento para la reutilización de la experiencia obtenida en la corrección de defectos de software", Paradigma - Revista Electrónica en Construcción de Software, Vol.2/ Num.3.

[74]. – Felipe Romero, "Administración de Conocimiento en Desarrollo de Software para Disminuir el Esfuerzo de Corregir Defectos", III Congreso Colombiano de Computación, Medellín - Colombia, 2008.

[75]. – M. Lenz, 1998, Defining knowledge layers for textual case-based reasoning. En Smyth y Cunningham (1998), páginas 298–309.

[76]. – Michel Jaczynski and Brigitte Trousse, An object-oriented framework for the design and the implementation of case-based reasoners, In Proceedings of the 6th German Workshop on Case-Based Reasoning, 1998.

[77]. – Michael M. Richter, The Knowledge Contained in Similarity Measures. Invited, Talk at the first International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra Portugal, 1995.